UNIVERSIDAD DE LOS ANDES, FACULTAD DE INGENIERÍA
DIVISIÓN DE ESTUDIOS DE POSTGRADO
PROGRAMA DE ESTUDIOS DE POSTGRADO EN COMPUTACIÓN
MÉRIDA- VENEZUELA

ALGORITMO HÍBRIDO PARA REALIZAR CLASIFICACIÓN Y AGRUPAMIENTO EN MINERÍA DE DATOS.

Autor: Ing. Fannia K. Pacheco M.

Tutores: Dr. Jose L. Aguilar C.
Dra. Mariela Cerrada L.
Dr. Junior Altamiranda P.

Trabajo de Grado presentado ante la ilustre Universidad de Los Andes como requisito parcial para optar al grado de *Magister Scientiae* en *Computación*.

Mérida, Febrero 2015

Algoritmo híbrido para realizar clasificación y agrupamiento en minería de datos.

Ing. Fannia K. Pacheco M.

Proyecto de Grado –Postgrado de Computación, 165 Páginas.

Escuela de Ingeniería de Sistemas, Universidad de Los Andes, 2015.

Resumen: La minería de datos (MD) está asociada al descubrimiento de información oculta en grandes cantidades de datos, siendo de gran interés en la actualidad, debido a la masiva cantidad de datos disponible hoy día en diferentes áreas. En la MD hay dos enfoques de aprendizaje para la obtención de patrones: el aprendizaje supervisado y el no supervisado, las tareas de MD comúnmente asociadas a dichos enfoques son de clasificación y agrupamiento. Normalmente, el aprendizaje supervisado trata con bases de datos conformadas por ejemplos etiquetados para la creación del modelo, mientras que el aprendizaje no supervisado utiliza ejemplos no etiquetados. En este trabajo se propone un algoritmo híbrido que puede trabajar con mecanismos de aprendizaje supervisado y no supervisado, tal que pueda ser usada en bases de datos con ejemplos etiquetados y no etiquetados. El diseño está inspirado en la construcción de grupos/clases basados en prototipos, utilizando el concepto de similaridad con el centroide y la estrategia el vecino más cercano 1-NN. Este tema es relevante debido a que los problemas de mayor interés para resolver con MD, están formados por la combinación de entradas etiquetadas y no etiquetadas. El algoritmo híbrido fue probado en bases de datos reales para resolver problemas de clasificación y agrupamiento, y se compararon los resultados con algoritmos clásicos, logrando un buen desempeño.

Palabras claves: minería de datos, algoritmos híbridos, clasificación, agrupamiento.

Índice

Índice de Tablas

Índice de Figuras

Capítulo 1

Introducción

La minería de datos (MD) es una herramienta que se caracteriza principalmente por permitir la extracción de conocimiento en grandes bases de datos, siendo de gran utilidad en una gran variedad de problemas, ya que en la actualidad se almacenan gran cantidad de datos, que crecen en una proporción considerable. Básicamente, consiste en una serie de procedimientos que permiten obtener patrones, asociaciones, estructuras significativas o modelos, a partir de los datos almacenados en bases de datos. El resultado obtenido provee conocimiento sobre los casos que están ocurriendo en el problema bajo estudio (Fayyad, U. et al, 1996).

En la MD hay dos enfoques de aprendizaje bien diferenciados para la obtención de patrones: el aprendizaje supervisado y el no supervisado. En el aprendizaje supervisado, los datos para realizar el entrenamiento del modelo de MD se definen como ejemplos, y los mismos están formados por las entradas al sistema y las salidas que se deberían obtener con esas entradas. En este caso, los ejemplos son comúnmente "entradas etiquetadas[1]". Por otro lado, en el aprendizaje no supervisado todo el proceso se lleva a cabo sobre un conjunto de ejemplos formados

[1]entradas etiquetadas son objetos que tienen una etiqueta representada con un valor (simbólico) discreto, que además, es conocido para cada uno de ellos.

tan solo por entradas al sistema, no se tiene información de cómo debería ser la salida. Por lo tanto, en este caso el sistema debería ser capaz de reconocer y construir patrones para poder etiquetar los nuevos ejemplos. En este caso, los ejemplos se suelen llamar "entradas no etiquetadas".

Uno de los principales problemas asociados al aprendizaje supervisado es la clasificación, en el cual se tienen entradas etiquetadas. En este caso, el algoritmo de aprendizaje define a que clase pertenece dicha entrada observando sus características (también llamados atributos), donde una clase se define como un conjunto de elementos que tienen características semejantes, que describen un determinado tipo de objeto. Así mismo, el aprendizaje no supervisado es a menudo referido a problemas de agrupamiento (clustering) de ejemplos no etiquetados, su objetivo es buscar relaciones entre ejemplos y construir grupos, para reunir ejemplos con características más similares.

Uno de los problemas de mayor interés para resolver con MD, consiste en extraer conocimiento desde fuentes de datos con entradas etiquetadas y no etiquetadas, es decir, definir criterios para resolver el problema híbrido de clasificación y agrupamiento.

En ese sentido, el algoritmo a proponer debe ser capaz de combinar los conceptos de aprendizaje supervisado y no supervisado, para lograr que en un mismo sistema se puedan realizar las tareas de clasificación (cuando la base de datos cuente con todos sus ejemplos etiquetados), agrupamiento (cuando la base de datos cuente con solo ejemplos no etiquetados), o híbrido (si la base de datos tiene ejemplos no etiquetados y etiquetados). El algoritmo debe contar con todas las funcionalidades de un algoritmo de MD clásico (sus fases de entrenamiento, prueba y funcionamiento), y además, de un módulo para la lectura y adaptación de los datos al sistema desarrollado. Este último módulo, se debe a que las bases de datos

comúnmente vienen en diferentes formatos, por ello, es necesario un procesamiento para su lectura.

1.1. Antecedentes

En la actualidad se han desarrollado una gran cantidad de algoritmos de MD para la extracción de conocimiento, ya sea para tareas de clasificación o agrupamiento, de manera separada. Muchos algoritmos se han desarrollado para resolver problemas específicos, o para mejorar el desempeño en las tareas de clasificación y agrupamiento, pero sin duda, uno de los retos más relevantes en la MD es la hibridación de tareas de clasificación y agrupamiento (Yang y Wu, 2006).

A continuación se presenta un estudio de los principales trabajos encontrados en la literatura relacionados a algoritmos híbridos de clasificación y agrupamiento, con la finalidad de observar las diferencias con nuestro planteamiento y conocer el avance en este tema en particular.

Un tipo de hibridación son los algoritmos supervisados mezclados, en los cuales se combinan reglas y árboles de decisión. Un ejemplo de estos algoritmos híbridos es el *Cover Learning using Integer LinearProgramming* o CLILP por sus siglas en inglés, además, de sus otras versiones CLILP2, CLILP3 y CLILP4 que introducen diferentes mejoras con respecto a la propuesta inicial CLILP (Cios at al. 2004,2001, 1995, 1995). La idea general se basa primero en la partición del conjunto de ejemplos, siguiendo una serie de condiciones que se aplican sobre los valores de los atributos de los datos, en subconjuntos de similares características en un árbol de decisión. Cada nivel del árbol es construido usando atributos que permitan distinguir entre la clase de un ejemplo y la clase de otro, dichos atributos son representados como nodos del árbol y permiten la creación de nuevas ramas que representan los posibles valores que puede tomar cada atributo. El segundo paso es tratar de seleccionar los nodos hoja más significativos, que se detectan utilizando

una métrica que permita contabilizar la información que aporta cada nodo, para luego seleccionar los nodos que aporten mayor información. Para cada uno de los nodos seleccionados se crean una serie de reglas basada en la serie de criterios en su rama. Finalmente, se seleccionan las mejores reglas que cubran la más alta cantidad de escenarios en la base de datos.

Existen otros algoritmos denominados "híbridos de clasificación", en los cuales se combinan los resultados de un problema de MD obtenidos con diferentes algoritmos de clasificación, para así obtener una mayor precisión. Debido a que en MD cada algoritmo de clasificación converge a una solución diferente, y es probable que no tenga un buen desempeño bajo ciertas condiciones, se utilizan diferentes algoritmos de clasificación para obtener los resultados de cada uno de ellos, este paradigma se basa en la concepción de que muchas mentes piensan mejor que una. Cada algoritmo utilizado en este tipo de técnica se considera como un "experto", por lo tanto se tienen varios expertos que dan su opinión, y por último, se utiliza un método para combinar estas opiniones. Los métodos utilizados para combinar los resultados dados por los expertos son (Breiman L.1996; Freund Y and Schapire R.,1999):

- *Bagging*: Dado un conjunto de datos de entrenamiento *D*, se crean *N* clasificadores usando la técnica *Boostrap* (Efron, B. y Tibshirani, R., 1993). *Boostrap* es una técnica utilizada para realizar un muestreo aleatorio con reemplazo sobre el conjunto de entrenamiento, con el resultado de ese muestreo se entrena cada clasificador.

  La salida de un nuevo ejemplo es por medio de votación, es decir, es aquella clase que obtuvo la mayoría de votos por los clasificadores.
- *Boosting*: Es un proceso iterativo que añade un clasificador en cada iteración, a cada ejemplo del conjunto de entrenamiento se le asigna un peso (inicialmente todos los pesos son iguales); se entrena el primer clasificador; se calcula el error, se incrementan los pesos de los ejemplos que fueron

clasificados erróneamente y se asigna un peso al clasificador en función de su precisión. Seguidamente se añade un nuevo clasificador y se entrena con los ejemplos con los pesos más altos, este procedimiento se repite hasta un número definido de iteraciones. La salida de un ejemplo viene generada por la votación ponderada por los pesos de todos los modelos.

- *Stacking*: Como primera etapa se utilizan diferentes tipos de clasificadores y se entrenan con los datos de entrenamiento, seguidamente se entrena otro modelo utilizando los resultados de los clasificadores de la primera etapa. Este último es el que genera la salida final (es un modelo por encauzamiento de resultados).

Chan *et al* (1999) utilizan los algoritmos de Naive Bayes, C4.5, CART, and RIPPER como clasificadores base, y usa la técnica de *stacking* para combinarlos. Phua *et al* (2004) proponen redes neuronales con retropropagación, naive Bayes y C4.5 como clasificadores base, y usa un único meta-clasificador para combinar las predicciones de esos clasificadores base (bagging), para producir el máximo ahorro de costes en las reclamaciones de seguros de automóviles.

A continuación analizamos otros algoritmos híbridos de clasificación, que se basan en el uso de entradas etiquetadas y no etiquetadas para resolver dicha tarea. Gabrys y Petrakieva (2002) presentan una generalización de los enfoques de diseño que normalmente siguen estos tipos de algoritmos. Los autores dividen las técnicas híbridas para combinar entradas etiquetadas y no etiquetadas en 3 categorías:

a. Enfoque Pre-etiquetado

   En este caso se permiten entradas etiquetadas y no etiquetadas para crear un sistema de clasificación. Este enfoque es muy utilizado ya que en la mayoría de las aplicaciones se consiguen más ejemplos no etiquetados que etiquetados. La idea principal de este enfoque es utilizar un clasificador para etiquetar los ejemplos no etiquetados al inicio del proceso, y luego entrenar.

Esta idea puede ser contraproducente en algunos casos, ya que si el etiquetado es inexacto en el primer clasificador, puede ocurrir que los datos correctos sean eclipsados por los nuevos datos etiquetados incorrectamente.

El procedimiento es el siguiente: En primer lugar, entrenar a un clasificador utilizando los datos con etiqueta. En segundo lugar, aplicarlo a los datos sin etiqueta para etiquetarlo en una clase (fase de "expectativa"). En tercer lugar, formar un nuevo clasificador utilizando las etiquetas de todos los datos (fase de "maximización"). En cuarto lugar, repetir hasta converja. Se podría pensar que es un agrupamiento iterativo (Witten I. et al., 2005).

Goldman y Zhou (2000) presentan una nueva estrategia que utiliza dos algoritmos supervisados diferentes que son originalmente entrenados con los datos etiquetados. Luego, se evalúan las instancias no etiquetadas en las dos hipótesis obtenidas de los algoritmos supervisados, cada uno genera la etiqueta que debería tener el ejemplo no etiquetado. Para tomar la decisión de cual etiqueta asignar al ejemplo, se debe cumplir que la precisión de una hipótesis sea mayor que la precisión del otra. La precisión se calcula obteniendo la validación cruzada de cada hipótesis y de cada clase. Estos enfoques normalmente se les llama *"co-training"* debido al entrenamiento inicial que se realiza.

b. Enfoque post-etiquetado

Se genera un modelo a partir de todos los datos, sin utilizar las etiquetas de los mismos, esto se puede alcanzar utilizando algoritmos no supervisados. Por lo general, se lleva a cabo mediante la aplicación de un procedimiento de estimación de la distribución de densidad de los datos o algoritmos de agrupamiento. Las etiquetas se utilizan posteriormente para el etiquetado de los grupos o de la estimación resultante, que implica el etiquetado de los datos.

c. Enfoque Semi-supervisado

Se define como un agrupamiento semi o parcialmente supervisado, en el que tanto datos etiquetados y no etiquetados se procesan a la vez. Este enfoque híbrido es el más utilizado en la actualidad.

En el trabajo desarrollado por Salama K. and Freitas A. (2013) se presenta un algoritmo semi-supervisado que divide la tarea en dos fases, la primera de agrupamiento y la segunda de clasificación. Se utiliza optimización basada en colonias de hormigas para la construcción de los grupos, ya que la misma permite asignar una nueva de entrada a un grupo, evaluando la feromona que se produce en cada grupo dada dicha entrada. Para la segunda etapa, se utiliza una red bayesiana múltiple (BMN) para realizar la clasificación, como se muestra en la Figura 1.1.

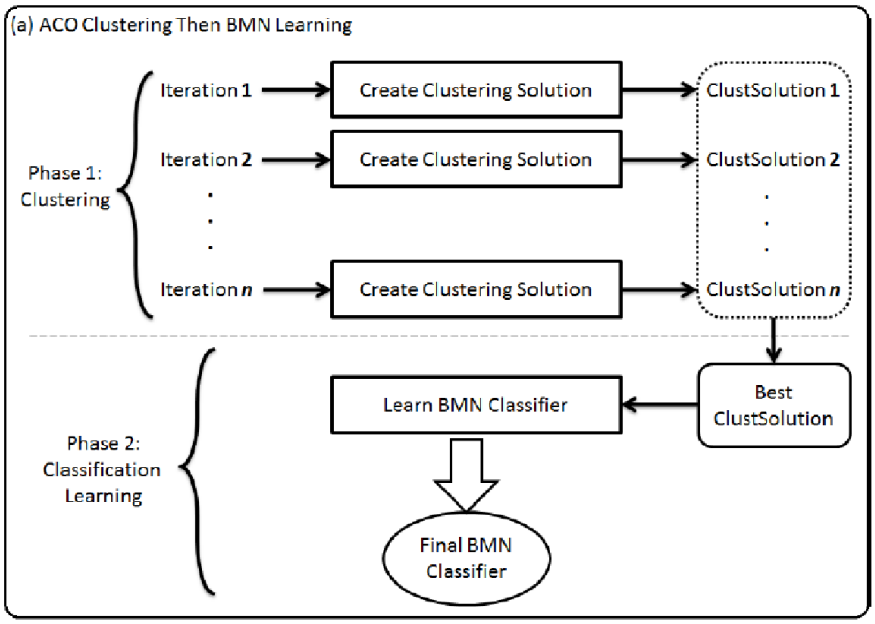


Figura 1.1. Representa el enfoque en dos fases "*cluster-then-learn*" para el aprendizaje de agrupamiento basada en clasificadores BMN (Salama K. and Freitas A., 2013).

Cabe destacar que no todas las técnicas híbridas pueden ser clasificadas como se mostró previamente, sin embargo abarca mucha de ellas. A continuación se muestran otros trabajos relacionados a técnicas híbridas.

Para realizar clasificación de imágenes es muy utilizada la MD. Pedrycz (2008) presenta un mecanismo para realizar supervisión parcial utilizando agrupamiento difuso, es decir, se plantea la creación de grupos que luego serán etiquetados por un experto. Este sistema ofrece una especie de guía al experto para realizar el etiquetado de las imágenes que fueron agrupadas, por ello, se denomina supervisión parcial. Para realizar el agrupamiento de las imágenes utiliza el algoritmo *fuzzy c-means* (FCM). El autor presenta un componente adicional expresando el nivel de coincidencia entre el grado de pertenencia producido por la FCM y la asignación de la clase que proporciona el usuario.

Karayiannis (1997) presenta una estructura para la construcción y entrenamiento de una red neuronal de base radial (RBF). La función de base radial creciente (GRBF) propuesta comienza con un pequeño número de prototipos que determina la ubicación de las funciones de base radial (grupos). En el proceso de entrenamiento, la red GRBF va creciendo por medio de la división de prototipos creados, mediante el uso de dos criterios de separación.

El enfoque propuesto permite que las redes GRBF crezcan durante el entrenamiento, aumentando gradualmente el número de prototipos que representan los vectores de características en el conjunto de entrenamiento, los cuales desempeñan el papel de los centros de las funciones de base radial. El proceso de aprendizaje se ilustra a grandes rasgos en la Figura 1.2. El aprendizaje comienza por determinar la mejor de dos particiones del espacio de características. Esto se puede lograr mediante el empleo de un algoritmo no supervisado o de uno supervisado. Un esquema de aprendizaje supervisado se emplea posteriormente para entrenar la red neuronal de base radial, si la salida no corresponde al valor deseado se dividen los prototipos siguiendo una serie de criterios, y se continua con este proceso hasta evaluar todos los ejemplos de la base de datos.

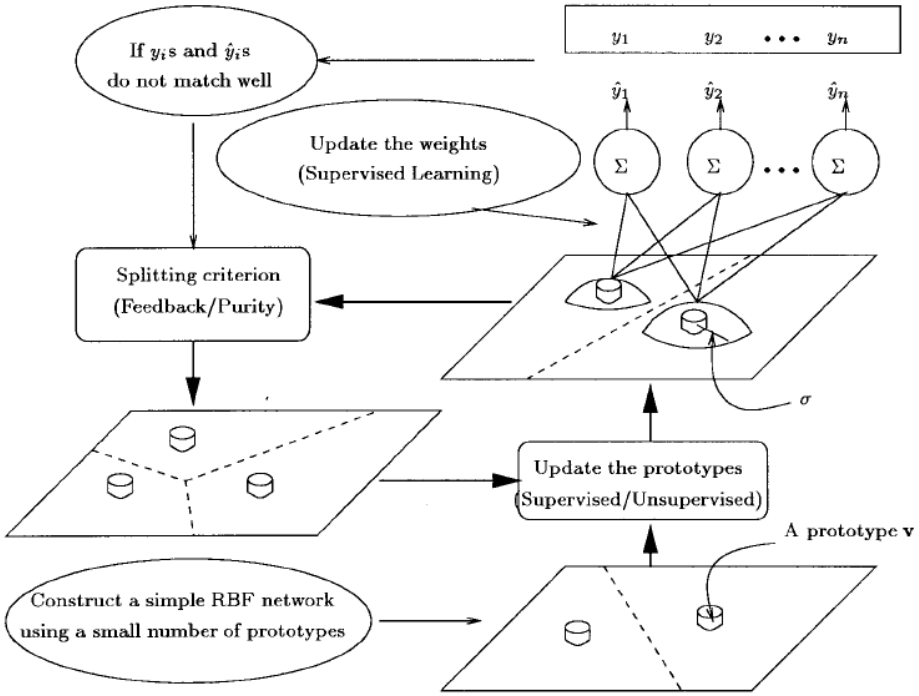


Figura 1. 2. El esquema de construcción y aprendizaje propuesto para las redes neuronales GRBF (Karayiannis N. and Weiqun G., 1997).

La estructura propuesta permite incorporar algoritmos de aprendizaje supervisado, así como también, algoritmos de aprendizaje no supervisado para el entrenamiento. No se especifica si la propuesta puede ser utilizada para agrupamiento y todas las pruebas fueron realizadas para problemas de clasificación. Se hace llamar un esquema híbrido de aprendizaje supervisado y no supervisado, sin embargo, no es completamente cierto ya que solo trabajan con problemas de clasificación.

Otro trabajo híbrido propuesto que se asemeja más a lo que plantea la presente investigación fue desarrollado por Gabrys (2000), que utiliza dos algoritmos propuestos por Simpson (1992 y 1993), el primero para realizar clasificación utilizando una red neuronal difusa, y el segundo para hacer agrupamiento también con una red neural difusa. En su trabajo, Gabrys (2000) con la unificación de estas estructuras (red neuronal difusa para clasificación y agrupamiento) plantea combinar aprendizaje supervisado con el no supervisado en un mismo algoritmo. La fusión de

agrupamiento y clasificación resultó en un algoritmo que puede ser utilizado como agrupación pura, clasificación pura, o como un sistema híbrido de clasificación y agrupamiento.

La red neuronal está compuesta por tres capas, los nodos en la segunda capa crecen adaptativamente ya que representan las clases y los grupos dado un problema de estudio. La entrada viene dada por el par $\{X_h, d_h\}$, siendo $X_h = [X_h^l, X_h^u]$ los atributos difusos de la entrada y d_h contiene la etiqueta de cada entrada en la posición h. Cada nodo en la segunda capa representa una clase o un grupo, debido a que están compuestos por una función de pertenencia de forma trapezoidal rectangular, que se construye por medio de un algoritmo de aprendizaje. Esos nodos se expanden y contraen (los rectángulos) cuando se asigna un nuevo ejemplo en uno de ellos, siguiendo una serie de criterios. Por último, los autores presentan una serie de pruebas usando bases de datos con solo entradas etiquetadas, no etiquetadas, y con la combinación de entradas etiquetadas y no etiquetadas. Este algoritmo no fue ampliamente probado en casos que contienen combinación de ejemplos etiquetados y no etiquetados en la misma base de datos.

Una estructura híbrida interesante semi-supervisada para realizar agrupamiento, se basa en dividir los datos no etiquetados en dos conjuntos. Un conjunto de los datos no etiquetados se etiqueta definiendo a que grupo pertenece utilizando cualquier algoritmo de agrupamiento. El otro conjunto de datos se utiliza para ser evaluado sobre el resultado del primer modelo, simulando el enfoque *co-training* pero para agrupamiento. Basu S. y su colaboradores (2004) plantea el uso de campos aleatorios ocultos de Markov (HMRF por sus siglas en inglés) para generar restricciones y crear un modelo generativo probabilístico para agrupamiento semi-supervisado. Es basado en una función objetivo que satisface restricciones, una función de distancia es construida y entrenada sobre el conjunto de datos etiquetados, satisfaciendo un conjunto de restricciones. Luego de entrenar el

modelo, la segunda fase es evaluarlo con el resto de los ejemplos (no etiquetados). HMRF-KMeans es un ejemplo de esta estructura, se utiliza K-Means para definir los centroides de los grupos, seguidamente se verifica que se cumplan las restricciones para estimar una nueva función de distancia. HMRF juega un papel importante, ya que se utiliza para generar las restricciones del modelo.

Para el reconocimiento de patrones en videos es muy utilizado el aprendizaje semi-supervisado, debido a la gran cantidad de videos etiquetados y no etiquetados disponibles. Yahong H. y sus colaboradores (2015) presentan un esquema semi-supervisado para el reconocimiento de patrones en videos, mostrado en la Figura 1.3, que se basa en varios teoremas matemáticos particulares sobre matrices. Básicamente, se utiliza una base de datos con videos etiquetados y no etiquetados, y para cada video en el conjunto de entrenamiento se extrae características de altas dimensiones para formar la matriz de características $X = [X_L, X_U]$; se define X_L como todos los videos etiquetados y X_U los videos no etiquetados. Se procesan los videos etiquetados X_L para codificar las etiquetas de los mismos en un nueva matriz. Seguidamente, para aprovechar la gran cantidad de videos no etiquetados se interpolan dichos videos en un diseño geométrico utilizando *splines*[2] *regression*. Como se muestra en la Figura 1.3, se separan las características del video y se agrupan las que tengan una geometría semejante. El siguiente paso es combinar el resultado de la interpolación con la matriz que contiene los videos etiquetados, y así definir las similitudes entre los videos etiquetados con las diferentes geometrías obtenidas de los videos no etiquetados. La meta final es obtener una matriz de transformación *W,* que sea capaz de seleccionar las características más resaltantes de los videos de prueba, y a su vez, prediga o genere los patrones encontrados.

[2]splines combinaciones polinómicas y funciones Green que son usadas para interpolar datos dispersos en un diseño geométrico (Adams R , 1975).

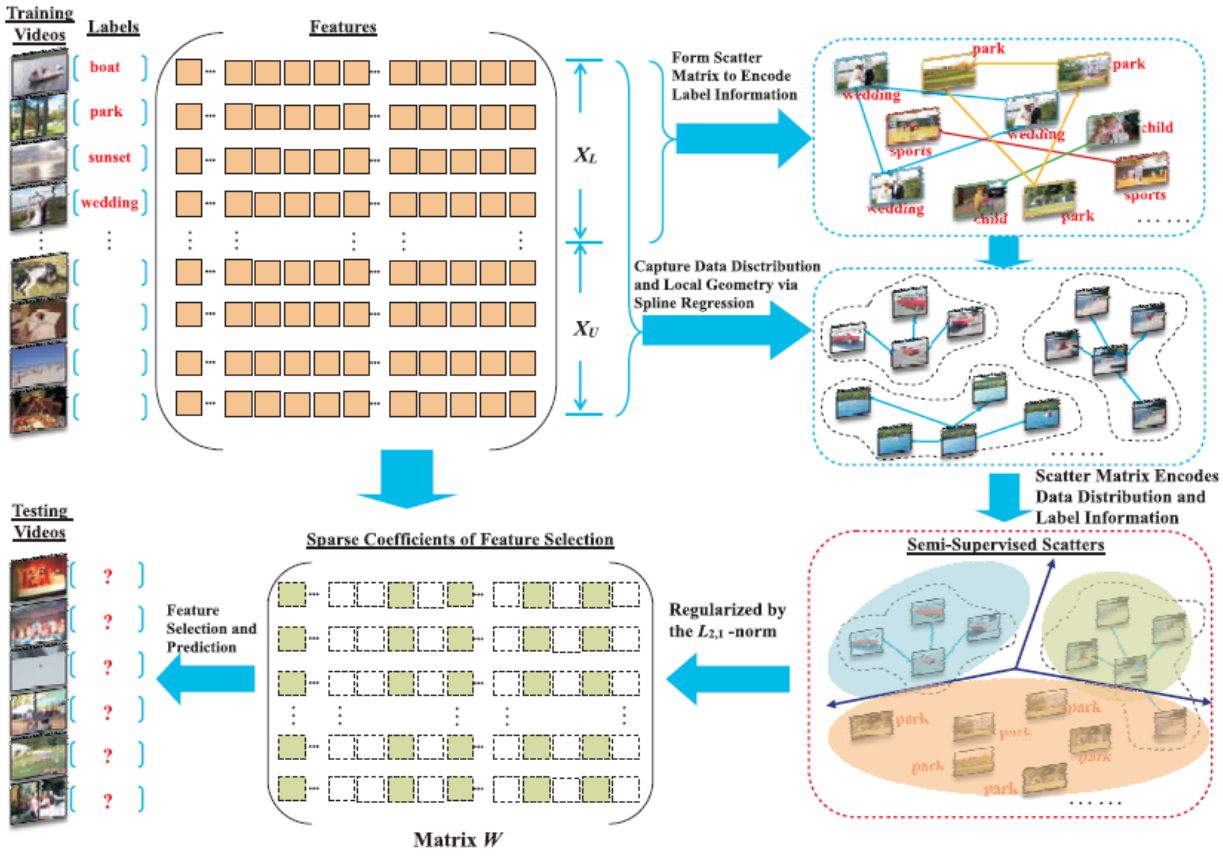


Figura 1.3. Esquema semi-supervisado de entrenamiento y prueba para el reconocimiento de patrones con el uso de videos etiquetados y no etiquetados (Tomada de Yahong H. et al, 2015).

1.2. Planteamiento del Problema

Específicamente, en el presente trabajo se presenta el diseño de un algoritmo de MD, que combinará la clasificación y el agrupamiento en un solo algoritmo. El problema de hibridación que se propone en esta investigación es la unificación del aprendizaje supervisado y no supervisado, es decir, que se permitan entradas etiquetadas y no etiquetadas en el sistema, y que el mismo responda de manera adecuada según el caso.

Dicho prototipo podrá realizar clasificación pura, agrupamiento puro, o un caso especial que lo denominamos híbrido, el cual está conformado por clases y grupos en el mismo modelo. Este último es de gran interés, ya que ha sido abordado solo por uno de los trabajos estudiados en los antecedentes de manera muy general.

1.3. Justificación

El principal problema es que muchas de las técnicas en MD son diseñadas para problemas específicos de clasificación o agrupamiento, pero hay muy pocas teorías unificadas, aunque si se consiguen gran cantidad de problemas que están conformados por entradas etiquetadas y no etiquetadas, por ejemplo: existe gran cantidad de imágenes y videos, especies en la Biología y otras áreas de la ciencia, etiquetados y no etiquetados, que implican constantes cambios en sus bases de datos. Este trabajo forma parte de uno de los retos a resolver en la minería de datos, siendo uno de los principales problemas en esa área de investigación, como lo presenta Yang (2005) en su revisión.

La mayoría de los problemas que se pueden resolver utilizando MD, cuentan con entradas etiquetadas y no etiquetadas, que normalmente utilizan un algoritmo semi-supervisado para realizar el etiquetado, pero que ocurre cuando estas entradas no pertenecen necesariamente a una etiqueta ya existente. Otro caso que se puede considerar es cuando se trabaja solo con aprendizaje no supervisado y comienzan a aparecer ejemplos etiquetados, y así como estos casos se consiguen diferentes variaciones, ¿qué hacer en estas situaciones en las que se combinan los conceptos relacionados al aprendizaje supervisado y no supervisado?

Algunos enfoques han tratado de resolver estas inconsistencias, como pudimos ver en los antecedentes, sin embargo los resultados han sido muy dependientes a los problemas a resolver o muy rígidos en algunos casos. Es por ello la motivación de esta investigación, con la finalidad de proponer un algoritmo flexible y unificado.

1.4. Objetivos

1.4.1. Objetivo General

Diseñar un algoritmo híbrido para realizar clasificación y agrupamiento en problemas de minería de datos

1.4.2. Objetivos Específicos

- Identificar las diferentes situaciones que se pueden presentar al unificar el aprendizaje supervisado y no supervisado, con la finalidad de definir un algoritmo híbrido para realizar clasificación y agrupamiento en problemas de minería de datos.
- Definir las estrategias que permitirán resolver los diferentes casos que se presentan en un algoritmo híbrido de este tipo.
- Seleccionar los métodos computacionales (técnicas y herramientas) adecuados que permitan la implementación del algoritmo.
- Desarrollar *benchmarcks* que permitan comparar el algoritmo híbrido con trabajos previos.

1.5. Estructura del Documento

El manuscrito está organizado de la siguiente manera:

Capítulo 2 describe las bases teóricas para el correcto desarrollo del trabajo de investigación.

Capítulo 3 define el diseño del sistema. Se define el diseño teórico del algoritmo híbrido, además, se presentan cada una de las etapas que conforman nuestra propuesta con sus respectivos macro-algoritmos.

Capítulo 4 presenta la implementación del algoritmo híbrido, las diferentes pruebas para validar la propuesta, y el análisis de los resultados obtenidos.

Capítulo 5, contiene las conclusiones del proyecto desarrollado y las recomendaciones para trabajos futuros, tomando como base el diseño propuesto.

Capítulo 2

Marco Teórico

Las bases teóricas expuestas ayudarán a entender el objetivo de la Minería de Datos (MD), así como también, los enfoques de aprendizaje supervisado y no supervisado utilizados en la MD. Debido a que el algoritmo a diseñar tiene como principio plantear una teoría que híbrida los conceptos de aprendizaje supervisado y no supervisado, específicamente las tareas de clasificación y de agrupamiento; a continuación se plantean sus bases teóricas y sus algoritmos más resaltantes. Adicionalmente, se estudiarán algunos algoritmos semi-supervisados y sus objetivos, para observar las diferencias con nuestra propuesta. En este capítulo también se explica brevemente el funcionamiento de Principal Components Analysis(PCA). PCA es ampliamente utilizado como una herramienta que permite la visualización de datos con altas dimensiones. Será útil para nuestra propuesta a fin de observar los resultados y las características de los datos gráficamente.

2.1. Minería de Datos

La MD surge de la necesidad de extraer conocimiento valioso de una gran cantidad de datos. Desde un punto de vista académico, el término "Minería de Datos" es una etapa dentro de un proceso mayor llamado extracción de conocimiento en bases de datos. Una definición tradicional de la minería de datos es: "*Proceso no trivial de*

identificación válida, novedosa, potencialmente útil y entendible de patrones comprensibles que se encuentran ocultos en los datos" (Fayyad, U. *et al.*, 1996). Para la extracción de conocimiento de bases de datos se suelen utilizar diferentes marcos metodológicos, en los cuales la MD es uno de los pasos más resaltantes (Rangel C. 2013).

MD es un término genérico que engloba resultados de investigación, técnicas y herramientas usadas para extraer información útil de grandes bases de datos. En MD los datos, de los cuales se desea extraer conocimiento, se representan como un conjunto de ejemplos; dichos ejemplos son situaciones que han ocurrido en el problema en estudio. La salida generalmente toma la forma de una predicción o clasificación de un nuevo ejemplo, sin embargo, se pueden conseguir otras representaciones como asociaciones entre características de un ejemplo. Las principales características y objetivos de la MD son:

- Explorar grandes bases de datos, que normalmente son historiales de varios años, donde se puede extraer conocimiento valioso.
- Generar una salida útil para el(los) usuario(s)
- Encontrar diferentes relaciones entre variables que pueden proveer más conocimiento, además de la salida final.

2.1.1. Tipos de Conocimiento en MD

Se tienen básicamente cuatro diferentes estilos de conocimiento en las aplicaciones de MD:

- Clasificación: como su nombre lo indica su principal tarea es clasificar una entrada dada sus características.
- Asociación: es el descubrimiento de relaciones entre las características (atributos) que conforman la base de datos, dichas asociaciones permitirán conocimiento valioso según sea el problema de estudio.
- Agrupamiento: En este caso se trata de agrupar objetos que no tienen una clase definida, y por lo tanto, cada agrupación está conformada por objetos semejantes entre sí. Una vez que los grupos han sido obtenidos,

se pueden observar las características de cada grupo, para por ejemplo, realizar un etiquetado de los mismos.

- Predicción: La predicción puede ser vista como un tipo de clasificación, la diferencia es que en la predicción el objetivo es predecir un estado futuro en lugar de un estado actual. La predicción consiste en determinar un valor futuro, un estado futuro, usando modelos que se construyeron a partir de la información histórica que describe el fenómeno bajo estudio. Clásicamente, la predicción está asociada a valores numéricos continuos dados como salidas de esta tarea, a diferencia de la clasificación más pensada en término de clases (etiquetas o valores nominales).

2.1.2. Ejemplo inicial de MD

- El problema del clima

  Para comprender las nociones básicas de la MD, se presenta a continuación un ejemplo clásico que permite caracterizar la esencia del problema. Cabe destacar que la base de datos, con las cuales se trabaja en MD; son extensas; sin embargo, entender el problema es primordial para definir la estrategia a seguir.

  El problema del clima es una tabla de datos pequeña, que nos permitirá definir cuáles son las características de una base de datos en general. Se trata de una base de datos donde se almacenó las condiciones del clima y si se jugó o no en un campo de tenis (Witten I. et al., 2005).

  Una base de datos está compuesta por una cantidad de atributos que caracterizan el problema y de instancias (también llamadas ejemplos). En el presente problema los atributos son: *Pronóstico, Temperatura, Humedad, Viento y Jugar*; mientras que las instancias son las filas dadas en la Tabla 1.1. La Tabla 1.1 se denomina *Vista Minable* (Pacheco M. 2014), y está compuesta por todas las variables del proceso y los datos a considerar en el estudio de MD.

En general, los valores que pueden tomar los atributos son nominales y numéricos; entiéndase por atributo nominal que solo está compuesto por nombres y nunca por números. En este caso, todos los atributos son nominales, y los posibles nombres o condiciones que pueden tomar cada uno son:

- *Pronóstico: soleado, nublado y lluvioso*
- *Temperatura : caliente, templado, fresco y frio*
- *Humedad: alta y normal*
- *Viento: falso y verdadero*
- *Jugar: no y si*

Uno de los objetivos que podría alcanzar la MD en este contexto, es predecir si se puede jugar o no dadas las características del clima. Por lo tanto, se podría decir que el atributo *Jugar* es la clase que se desea predecir o clasificar.

Creando simples reglas decisivas se puede generar un modelo que permita realizar dicha predicción. Por ejemplo:

Si Pronóstico = soleado y humedad = alta, entonces jugar= no

Si observamos la Tabla 1.1 y la regla previamente definida, se puede observar que siempre que sea soleado y la humedad es alta, la decisión que se toma es no jugar. Mientras se van generando más reglas donde todos los casos sean cubiertos, va emergiendo la extracción de conocimiento, hasta conseguir un modelo (basado en reglas en este caso), que permita la predicción de si jugar o no.

Tabla 2.1. Base de datos del clima (Witten I. et al., 2005).

Pronóstico	Temperatura	Humedad	Viento	Jugar
soleado	caliente	alta	falso	no
soleado	caliente	alta	verdadero	no
nublado	caliente	alta	falso	si
lluvioso	templado	alta	falso	si
lluvioso	fresco	normal	falso	si
lluvioso	fresco	normal	falso	si
nublado	fresco	normal	verdadero	si
soleado	templado	alta	falso	no
soleado	fresco	normal	falso	si
lluvioso	templado	normal	falso	si
soleado	templado	normal	verdadero	si
nublado	templado	alta	verdadero	si
nublado	caliente	normal	falso	si
lluvioso	templado	alta	verdadero	no

Al observar la Tabla 1.1, es evidente que los valores de *Temperatura* y *Humedad* se pueden representar como valores numéricos, cambiando así las reglas para la predicción de la clase *Jugar*.

2.1.3. Entradas y salidas de un problema de MD

Hay tres definiciones asociadas a un problema de MD, que son importantes conocer (Witten I. et al., 2005):

- Instancias

  Las entradas al modelo de conocimiento son un conjunto de instancias (también denominadas ejemplos). Las instancias son las cosas que serán clasificadas, asociadas o agrupadas. Cada uno se caracteriza por los valores de un conjunto de atributos predeterminados.

- Atributos

  Cada instancia individual, que es una entrada del algoritmo de MD, es caracterizada por valores en un arreglo. Cada posición del

arreglo es caracterizado o definido como un atributo de dicha instancia. En consecuencia, una base de datos está compuesta por *N* atributos (columnas del arreglo) o conjunto de características. Los atributos pueden tomar valores nominales o numéricos, como se dijo antes, un atributo nominal está compuesto por nombres y un atributo numérico solo por números.

En el ejemplo descrito del clima de la Tabla 2.1, cada conjunto de datos se representa como una matriz de instancias (filas de la matriz) contra atributos (columnas de la matriz), que en términos de base de datos es una relación única, o un archivo plano.

- Salida: Representación del conocimiento.

  La mayoría de las técnicas de MD están destinadas a producir patrones, que puedan generar descripciones comprensibles del comportamiento de los datos. Dichos patrones descubiertos, pueden ser expresados de diferentes formas, eso lo determina el tipo de técnica que se está utilizando. Por ejemplo, la regla generada en el ejemplo del clima: "*Si Pronostico = soleado y humedad = alta, entonces jugar= no*", representa un patrón extraído de la base de datos.

2.1.4. Tipos de aprendizaje

Todo algoritmo de MD de datos está compuesto por un conjunto de entrenamiento, del cual se extrae información por medio de un algoritmo de aprendizaje (se desarrolla un modelo). Dicho modelo será usado para nuevas entradas y generará la salida del producto de MD. Este proceso se muestra en la Figura 2.1.

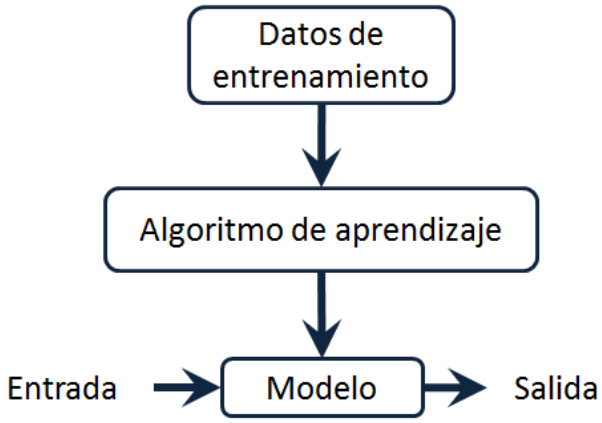


Figura 2. 1. Proceso de aprendizaje en MD.

Clásicamente, el algoritmo de aprendizaje puede ser desarrollado por medio de alguno de los dos siguientes enfoques, dependiendo de la entrada y el objetivo que se desea alcanzar con MD:

- Aprendizaje Supervisado

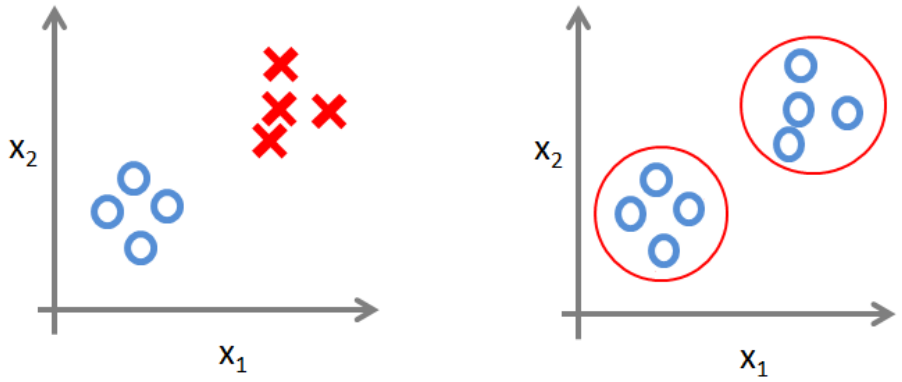
  El proceso de modelado se realiza sobre un conjunto de ejemplos formado por entradas al sistema y las respuestas que se deberían dar para cada entrada.

  Las tareas de clasificación, asociación y predicción son desarrolladas por medio de un aprendizaje supervisado. Cabe destacar que, la tarea de clasificación también puede realizarse por medio de un modelado no supervisado, pero con menos precisión.

- Aprendizaje no Supervisado

  Todo el proceso de modelado se lleva a cabo sobre un conjunto de ejemplos formado tan sólo por entradas al sistema. No se tiene información sobre las categorías de esos ejemplos. Por lo tanto, en este caso, el sistema tiene que ser capaz de reconocer patrones para poder etiquetar las nuevas entradas a través de procesos heurísticos e inductivos, para obtener como resultado: análisis, agrupamientos y dependencia de los objetos. La tarea de MD de agrupamiento es desarrollada por medio de aprendizaje no supervisado.

En la Figura 2.2 se puede observar gráficamente la diferencia entre aprendizaje supervisado y no supervisado. En la Figura 2.2-(a) se tienen bien definidas las formas de cada objeto, y por lo tanto pueden clasificarse como dos clases, una en forma de círculo y otra en forma de cruces. Mientras que en la Figura 2.2-(b) no se tienen definidas las clases de los objetos, solo se conocen las semejanzas entre ellos, por lo tanto se realizan dos agrupamientos.



(a) (b)

Figura 2.2. (a) Ejemplo gráfico de aprendizaje supervisado (b) Ejemplo gráfico de aprendizaje no supervisado.

2.2. Clasificación

El propósito de la clasificación es obtener una función o modelo que determine la clase de un objeto basado en las características de sus atributos. Para generar dicho modelo o función, es necesario definir un conjunto de datos de entrenamiento, el mismo está compuesto por objetos que ya tienen su clase asignada, también denominados ejemplos etiquetados.

El modelo o función es creado analizando las relaciones entre los atributos de los objetos y las clases en el conjunto de entrenamiento.

Mientras más variedad de escenarios se presenten en el conjunto de entrenamiento, más se enriquecerá el modelo de clasificación, generando mejores resultados en la clasificación de nuevas entradas no etiquetadas.

2.2.1. Algoritmos de Clasificación

Existen diferentes marcos teóricos que han permitido el diseño de algoritmos de clasificación, los mismos utilizan conceptos de estadística, inteligencia artificial, matemática, análisis de datos, etc. Por ello, se presenta a continuación las vertientes más comunes usadas en los algoritmos tradicionales, para el diseño de nuevos algoritmos de clasificación.

2.2.1.1. Algoritmos basados en análisis estadísticos

Estos algoritmos utilizan teorías de la estadística para realizar asociaciones entre los atributos. Uno de los algoritmos más conocidos es la clasificacion por medio del teorema de Naive Bayes (Christopher D. et al, 2008).

- Clasificacion Bayesiana

  Estos algoritmos utilizan el teorema de Bayes para la toma desiciones. En inferencia estadística, se trata de estudiar la distribución estadística de los datos examinando los valores de los atributos. Dado un conjunto de datos $X = \{x_1, x_2, \dots, x_n\}$, un problema de MD es descubrir las propiedades de la distibución de donde el conjunto viene. El teorema de Bayes es utilizado para estimar la probabilidad de que una hipótesis sea cierta dada una instancia. Entendiéndose como hipótesis la pertenencia a una clase.

  Dicho teorema viene dado por la ec.(2.1) donde $P(h_1|x_i)$ es la probabilidad de que la hipótesis h_1 sea verdadera dada el ejemplo x_i:

$$P(h_1|x_i) = \frac{P(x_i|h_1)P(h_1)}{P(x_i|h_1)P(h_1) + P(x_i|h_2)P(h_2)} \qquad (2.1)$$

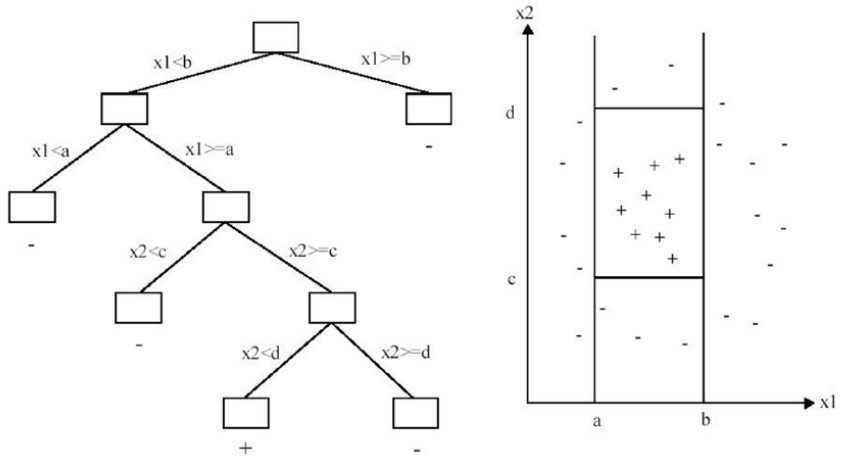
Entre otros algoritmos basados en análisis estadístico se encuentran:

- Gaussian Naive Bayes, Bernoulli Naive Bayes (Pedregosa F. et al. 2011)
- La regresión y sus variantes: regresión lineal, regresión logística, isotonic regresión, entre otros(George A. et al. ,2003).
- Procesos gaussanianos(Naren R. et al. ,2005)
- Redes bayesianas (Richard E., 2004)

2.2.1.2. Algoritmos basados en Árboles de decisión

Este algoritmo se basa en la construcción de un árbol, donde los nodos son los atributos y las hojas las clases. La representación del conocimiento es una de las más sencillas, para clasificar instancias con un número finito de clases.

En la Figura 2.3 se observa de manera general su planteamiento. Se trata de dividir el conjunto de ejemplos según ciertas condiciones que se aplican a los valores de las características. En la Figura 2.3-(a) se observa la construcción del árbol, mientras que el la Figura 2.3-(b) se observa graficamente la división de los ejemplos.



(a) (b)

Figura 2.3. (a)Construcción del árbol de decisión (b) División de los ejemplos en un plano bidimensional (Dunham M., 2003).

Para la construcción del árbol comunmente se utiliza la entropía como métrica para tomar la decisión de cómo particionar el árbol, ya que indica la medida de información que proporciona cada atributo. La entropía viene dada por la ec.2.2, donde *c* indica el número de clases existentes. $p(i|x_j)$ es la fracción de ejemplos pertenecientes a la clase *i* dado x_j , donde *j* la cantidad de atributos.

$$E(x_j) = -\sum_{i=1}^{c} p(i|x_j) log_2 p(i|x_j) \qquad (2.2)$$

Intrínsecamente, la entropía nos indica los atributos que aportan más información, y que por lo tanto son los nodos cercanos a la raiz, mientras que los atributos menos relevantes están cercanos a las hojas.

Existen otras métricas que permiten realizar la división del árbol, como el indice de Gini o el error de clasificación (Tan P., et al. 2005).

Si tomamos el ejemplo del clima (sección 2.1.3), gráficamente el primer paso se observa en la Figura 2.4-(a). Inicialmente todos los atributos son posibles candidatos y se selecciona el que aporte la medida de información más alta para particionar el árbol (en este caso *Pronóstico*). El atributo *Pronóstico* es seleccionado como nodo raiz, para

la siguiente iteración se realiza el mismo procedimiento con los atributos restantes, donde las configuraciones posibles se muestran en la Figura 2.4-(b). La condición de parada viene dada por ciertas restricciones que debe cumplir el algoritmo, basadas en métricas que describen el comportamiento del proceso de clasificación (normalmente, covertura mínima y precisión), finalmente se obtiene el árbol resultante dado en la Figura 2.4-(c).

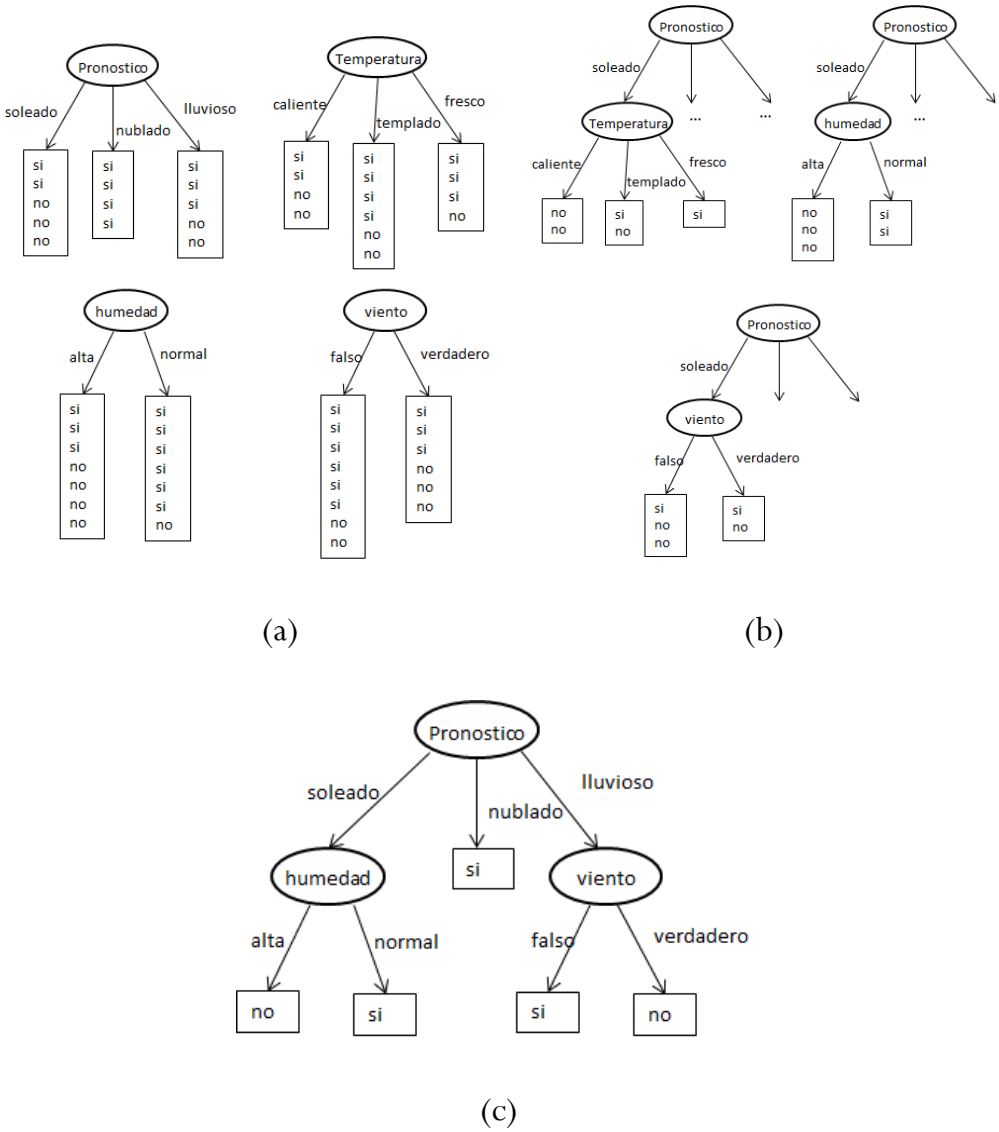


Figura 2.4. (a)Construcción del árbol de decisión primera iteración (b) Construcción del árbol de decisión segunda iteración (c)Construcción del árbol de decisión resultado final(tomado de Witten I. et al., 2005).

Existe una gran variedad de algoritmos basados en árboles de decisión, como por ejemplo: J48, CART, C4.5, ADtree,randomTree, REPTree, entre otros (Zhao Y. and Zhang Y. 2000**,** Breiman L. et al, 1984; Dunham M., 2003; Brigham A. and Andrew M. 1998; Breiman L. 2001).

2.2.1.3. Algoritmos basados en reglas

Es una de las estrategias más rudimentarias para realizar clasificación, en la cual se buscan asociaciones entre los attributos para construir reglas de clasificación. En el caso de que nuevas instancias sean evaluadas, solo se deben verificar las reglas desarrolladas y clasificar donde lo definan dichas reglas.

Tomando el ejemplo del clima (sección 2.1.3), una regla evidente alli es:

Si Pronóstico = lluvioso y viento = verdadero, entonces jugar = no

Las reglas de clasificación se pueden construir por medio de diferentes algoritmos. Los algoritmos más comunes son basados en árboles de decisión, en los cuales primero se construye el árbol de decisión y luego se extraen las reglas, sin embargo, existen algoritmos donde no es necesario la construcción del árbol, un ejemplo de este es *1-Rule* (Witten I. et al., 2005).

El algoritmo 1-Rule crea el primer nivel de un árbol de decisión. Aunque es un algoritmo simple ha demostrado ser efectivo en diversos problemas de MD, principalmente donde se involucran atributos nominales. El algoritmo crea una regla para cada atributo en el conjunto de entrenamiento, seguidamente selecciona la regla con el error más pequeño. Para crear una regla para un atributo, se determina

la clase más frecuente por cada valor del atributo. El pseudo codigo de este algoritmo se muestra en la Figura 2.5. Otro enfoque para la construcción de patrones por medio del uso de reglas, se basa en determinar una serie de reglas base, y las mismas se van combinando hasta que se minimiza el error.

```
For each attribute A,
   For each value VA of the attribute, make a rule as
   follows:
      count how often each class appears
      find the most frequent class Cf
      create a rule when A=VA;class attribute value = Cf
   End For-Each
   Calculate the error rate of all rules
   End For-Each
Chose the rule with the smallest error rate
```

Figura 2.5. Algoritmo de 1-rule.

Entre los algoritmos basados en reglas se encuantran: ZeroR, M5Rule, ConjunctiveRule, PART, entre otros (Mark H.et al, 2009).

2.2.1.4. Algoritmos basados en distancia

El principio de los algoritmos basados en distancia viene definido por la distancia entre ejemplos, donde una distancia pequeña podría significar alta similaridad y viceversa. Por lo tanto, para realizar el entrenamiento de dichos algoritmos, se ordenan los ejemplos que están más cercanos entre sí. Seguidamente, se utiliza aprendizaje supervisado tomando la etiqueta de los ejemplos para la creación de las clases.

Una vez definidas las clases, se procede a evaluar los ejemplos con etiquetas desconocidas, donde los mismos se asignan a las clases con la distancia más pequeña o similaridad más alta. Dicha evaluación se detemina utilizando una función de distancia o similaridad. Existen

diversas funciones de distancia, a continuación se presentan las más conocidas.

- Funciones de distancia

Dados dos vectores *n*-dimensionales (atributos), que describen dos instancias del conjunto de datos, de la forma:

$$u = \{u_1, u_2, \dots, u_n\}$$

$$v = \{v_1, v_2, \dots, v_n\}$$

La distancia entre *v* y *u* se calcula utilizando alguna de las métricas dadas en la Tabla 2.2. La selección de la métrica depende directamente de la estructura de los datos y de las caracteristicas propias de la métrica, por ello, se debe realizar un análisis previo para determinar la mejor métrica según el problema. Sin embargo, la distancia euclideana ha desmostrado ser una de las más efectivas en diversidad de problemas.

Tabla 2.2. Algunas métricas para calcular la distancia entre dos vectores (Xu R. y Wunsch D., 2005).

Métrica	Forma	Caracteristicas
Minkowski	$D(u,v) = \left(\sum_{i=1}^{n} \lvert u_i - v_i \rvert^p\right)^{1/p}$	Atributos con grandes valores y grandes varianzas tienden a dominar sobre otros atributos
Euclideana	$D(u,v) = \sum_{i=1}^{n} (u_i - v_i)^2$	Es una de las métricas en MD más utilizadas, es un caso especial de Minkowski con *p=2*. Corresponde a la longitud del camino más corto entre dos puntos
City-Block	$D(u,v) = \sum_{i=1}^{n} \lvert u_i - v_i \rvert$	Caso especial de Minkowski con *p=1*. Es conocida como la distancia de Manhattan. Es la suma de las distancias a lo largo de cada dimensión.

Continua en la próxima página …

Viene de la página anterior …

Mahalanobis	$D(u,v) = (u_i - v_i)^T S^{-1}(u_i - v_i)$ S^{-1} es la matriz de covarianza de los datos	Cuando los atributos no están correlacionados, la distancia cuadrática de Mahalanobis es equivalente a la distancia cuadratica euclideana. Puede causar mayor costo computacional.
Pearson	$D(u,v) = (1-r)/2$ $r = \sum_{i=1}^{n} \left(\frac{u_i - \bar{u}}{\sigma_x}\right)\left(\frac{v_i - \bar{v}}{\sigma_y}\right)$ Donde $\bar{v}$ y $\bar{v}$ son las medias de u y v respectivamente. σ_x y σ_y son las desviaciones estándar de u y v	Derivada del coeficiente de correlación de Pearson (Spiegel, M., 1992). Como el coeficiente de correlación toma valores entre [-1,1] para indicar el tipo de correlación, la distancia de Pearson se encuentra entre [0,2], y se interpreta igual que el coeficiente de correlación.
Cosine	$D(u,v) = \frac{u \cdot v}{\|u\|\|v\|}$	Es independiente del tamaño del vector. Mide la similaridad entre dos vectores *u* y *v*, calcula el coseno del ángulo entre ellos. Dos vectores con la misma orientación tienen una similaridad de 1, dos vectores separados por 90° tienen similaridad de 0. Esta métrica es usada en espacios positivos donde toma valores entre [0,1]

- Vecino más cercano

El algoritmo Vecino-más-cercano (o K-NN por sus siglas en inglés), es uno de los algoritmos más simples y eficases desarrollados para realizar clasificación basada en distancia, aunque su mayor debilidad es que se vuelve un proceso lento a medida que aumenta la cantidad de ejemplos a evaluar.

En este enfoque, los datos son representados en un plano n-dimensional, basicamente se trata de detectar quien es(son) el(los) vecino(s) más cercano(s) a un ejemplo, calculando la distancia del mismo a todos los ejemplos del conjunto de entrenamiento. Por lo

tanto, el tiempo que se toma en buscar el(los) vecino(s) más cercano(s) para realizar una clasificación es proporcional al número de instancias del conjunto de entrenamiento (Larose, D., 2004).

Se debe indicar a priori el número de clases que se van a crear dado un conjunto de entrenamiento, y la cantidad de vecinos *K* más cercanos que permiten decidir a que clase se asigna una nueva entrada. En el ejemplo dado en la Figura 2.6 se muestra una configuración de las clases, despues de buscar sucesivamente los vecinos de cada ejemplo. Normalmente *K* es igual a 1, asi un nuevo ejemplo sera asignado a su unico vecino más cercano.

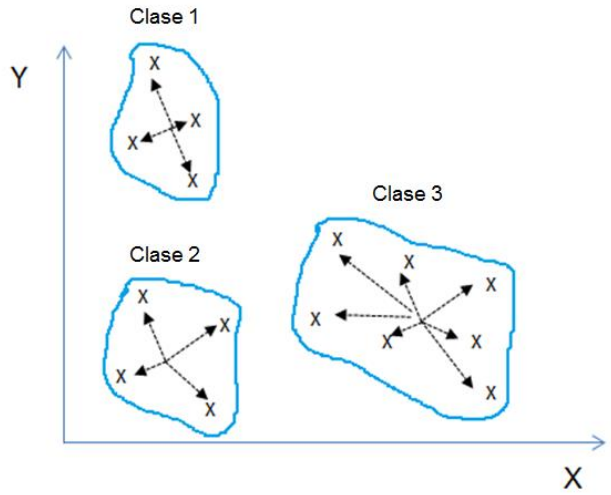


Figura 2.6. Ejemplo de configuracion de las clases usando el vecino más cercano.

El algoritmo de K-NN toma como entrada el conjunto de entrenamiento *T* y el número de vecinos *K* más cercanos. Este algoritmo toma un ejemplo del conjunto de entrenamiento y lo compara con todos los demás ejemplos, para asi detectar los *K* vecinos más cercanos. Los pasos más resaltantes para desarrollar este método son:

1. Dada la entrada $x \in T$

2. Encontrar los *K* vecinos más cercanos sobre los datos de entrenamiento $x_1, x_2, \ldots, x_n$ a x, dada una métrica de distancia previamente definida.
3. Dado *K* vecinos más cercanos al punto x, se hace una votación y se asigna el punto x a la clase que obtenga la votación más alta.

K-NN puede ser utilizado tanto para clasificación como para agrupamiento, es decir puede utilizar aprendizaje supervisado y no supervisado.

Un caso especial ampliamente utilizado es 1-NN, en este caso se asigna el nuevo ejemplo al vecino más cercano intrínsecamente sin realizar una votación. Es principalmente utilizado para el fortalecimiento de otros algoritmos supervisados y no supervisados.

Entre los algoritmos basados en distancia se encuentran: Nearest Centroid Classifier, Nearest Shrunken Centroid, Nearest Neighbors Regression (Pedregosa F. et al. 2011).

2.2.1.5. Algoritmos basados en redes neuronales

Se basan en el uso de una red neuronal artificial para realizar aprendizaje supervisado, los modelos comúnmente usados son el perceptron simple y el perceptron multicapa.

Básicamente, se plantea una hipótesis $h_\Theta(x)$, y dada una cantidad finita de estímulos $x_1, x_2, \ldots, x_n$, se genera una salida que permite concluir si la hipótesis definida es verdadera o falsa. Un modelo de una red neuronal con tres capas se muestra en la Figura 2.7, donde $a_i^{(j)}$ es la activación de la unidad i en la capa j y $\theta^{(j)}$ es la matriz de pesos de la red.

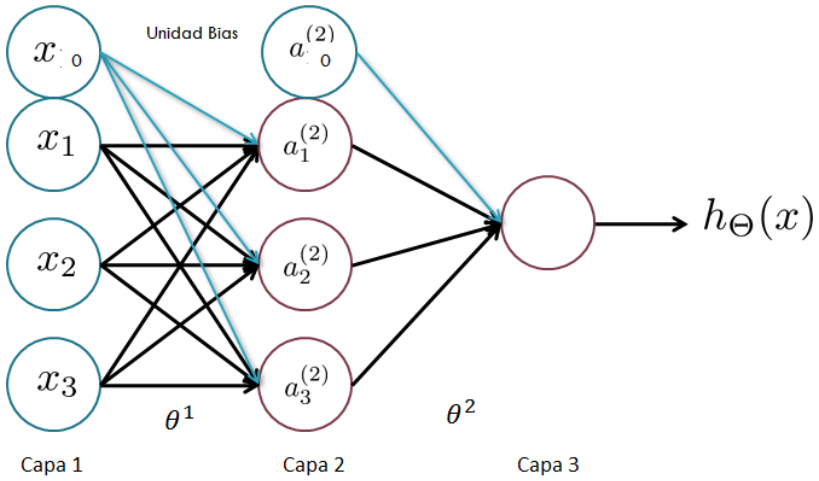


Figura 2.7. Red neuronal con tres capas.

Para el aprendizaje de la red neuronal se utiliza la retropropagación, que es un método basado en el gradiente descendiente para encontrar el error en una red hacia adelante, en el cual se necesita un conjunto de entrenamiento y sus valores esperados. Los pasos más resaltantes para el entrenamiento de una red neuronal para clasificación son:

a. Inicializar los pesos de la red aleatoriamente
b. Calcular $h_\Theta(x)$ para cada x
c. Para cada salida de la red neuronal, se calcula el error $\delta_i^{(j)}$ del nodo i en la capa j utilizando la ec.2.3, asumiendo que la salida del nodo i es d_i y y_i es el valor esperado.

$$\delta_i^{(j)} = d_i - y_i \qquad (2.3)$$

d. Se realiza entonces el paso hacia atrás, donde se utiliza el error calculado en la capa de salida para cambiar los pesos de cada capa oculta de la red neuronal. Una fórmula para dicha actualización se denomina la regla delta, y viene dada por la ec. (2.4). λ es la taza de aprendizaje de la red y x_{ij} las entradas de la capa j.

$$\Delta\theta^{(j)} = \lambda x_{ij}(d_i - y_i) \qquad (2.4)$$

Existen diversidad de algoritmos basados en redes neuronales, entre los más comunes se encuentran: RBFnetwork, BayesNetwork, Self-Organizing Maps, entre otros (Mark H.et al, 2009; Pedregosa F. et al. 2011; Hastie T. et al, 2009).

2.2.1.6. Algoritmos híbridos

La combinación de diferentes algoritmos de clasificación para resolver algunos problemas de MD, ha demostrado tener mayor exactitud con respecto a los clasificadores tradicionales. Este tipo de clasificadores son llamados híbridos o aclopamiento de clasificadores. Se basa en que cada clasificador es un experto, y da su opinion acerca del problema de clasificación. Los criterios para combinar clasificadores son:

- Votación por mayoría: Varios clasificadores votan para seleccionar la clase a la cual pertenece un ejemplo.
- Votación ponderada: Se pondera la votación de cada clasificador (en función de su precisión), para definir la clase a la cual pertenece un ejemplo.

Como vimos en el Capítulo 1, existen tres esquemas para combinar los resultados de los clasificadores: Bagging, Bosting y Stacking. Toda esa fase se denomina metaclasificador, la cual es la encargada del conteo de los votos y de tomar la decisión final.

2.2.2. Métricas para evaluar un algoritmo de clasificación

Para saber que tan bien funcionan los algoritmos de clasificación, existen diversos métodos de evaluación de los modelos obtenidos. Normalmente, se divide el conjunto de datos en un conjunto de entrenamiento y un conjunto de prueba. Esto es válido cuando se esta trabajando con bases de datos grandes. El conjunto de prueba definirá el

rendimiento del modelo, utilizando criterios de medición como error o precisión.

Dado que se conocen las clases reales a las que pertenece cada ejemplo del conjunto de prueba, se puede evaluar dichos ejemplos en el modelo y obtener la matriz de confusión dada en la Tabla 2.3. En la Tabla 2.3 se contabiliza:

- La cantidad de ejemplos de prueba clasificados correctamente en su clase (verdaderos positivos (tp)).
- La cantidad de falsos negativos (fn), los cuales representan los ejemplos clasificados como negativos, pero que realmente eran positivos en los ejemplos de prueba.
- La cantidad de ejemplos que fueron clasificados como positivos, que realmente son negativos (los falsos positivos (fn)).
- Por último, se contabilizan los ejemplos que son clasificados negativos correctamente, denominados verdaderos negativos(tn).

Este ejemplo específico tiene solo dos clases: Positivos y Negativos, en caso de tener más clases la tabla se extiende.

Tabla 2.3. Matriz de confusión con dos clases

Clasificación	Clasificados positivos	Clasificados negativos
Pos	Verdaderos positivos (tp)	Falsos negativos (fn)
Neg	Falsos positivos (fp)	Verdaderos negativos (tn)

Observando la tabla anterior, se puede llegar a varias conclusiones acerca del funcionamiento del modelo. Es por ello que se definen varias medidas de precisión utilizando los datos de la matriz de confusión (Sokolova M. y Lapalme G., 2009):

i. Precisión: número de ejemplos clasificados positivos correctamente entre el total de ejemplos clasificados como positivos por el modelo.

$$p = \frac{t_p}{t_p + f_p} \qquad (2.5)$$

ii. Recall: es el número de ejemplos clasificados positivos correctamente entre el total de ejemplos positivos en la base de datos.

$$r = \frac{t_p}{t_p + f_n} \qquad (2.6)$$

iii. Fscore: es una combinación de la precisión y recall.

$$Fscore = 2\frac{p * r}{p + r} \qquad (2.7)$$

iv. Error: se calcula con la precisión utilizando la ec.(2.8).

$$e = 1 - p \qquad (2.8)$$

Las métricas definidas previamente son las más comunes para la evaluación de modelos de clasificación. En caso de tener una multiclasificación las mismas se generalizan, como se muestra en la Tabla 2.4, donde μ y *M* el promedio micro y macro de los datos, y *n* el número de clases. La media macro trata a todas las clases por igual, mientras que la media micro favorece a las clases más grandes.

Tabla 2.4. Métricas para la evaluación de un modelo de clasificación (Sokolova M. y Lapalme G., 2009).

Métrica	Formula
Tasa de error	$\frac{\sum_{i=1}^{n} \frac{tp_i + tn_i}{tp_i + tn_i + fp_i + fn_i}}{n}$
Precision_{μ}	$\frac{\sum_{i=1}^{n} tp_i}{\sum_{i=1}^{n}(tp_i + fp_i)}$
Recall_{μ}	$\frac{\sum_{i=1}^{n} tp_i}{\sum_{i=1}^{n}(tp_i + fn_i)}$
Fscore_{μ}	$\frac{(\beta^2 - 1)presicion_{\mu} * recall_{\mu}}{\beta^2(presicion_{\mu} + recall_{\mu})}$
Precisión_{M}	$\frac{\sum_{i=1}^{n} \frac{tp_i}{tp_i + fp_i}}{n}$
Recall_{M}	$\frac{\sum_{i=1}^{n} \frac{tp_i}{tp_i + fn_i}}{n}$
Fscore_{M}	$\frac{(\beta^2 - 1)presicion_{M} * r}{\beta^2(presicion_{M} + r)}$

2.3. Agrupamiento

Como su nombre lo indica, esta tarea de MD se basa en un proceso de agrupamiento (o clustering, en inglés) de ejemplos con similares características, con la finalidad de encontrar patrones útiles que puedan proporcionar información de un problema en estudio.

Un grupo (ó cluster) está conformado por un conjunto de objetos similares entre sí, diferentes a otros conjuntos de objetos. Para definir qué es similar y disimilar en clustering, normalmente son utilizadas métricas de distancia como las dadas en la Tabla 2.5.

2.3.1. Algoritmos de agrupamiento

Al igual que los algoritmos de clasificación, comúnmente los algoritmos de agrupamiento son divididos en diferentes vertientes teóricas para el diseño

de sus algoritmos. Aqui se presentan las más relevantes para la presente investigación.

2.3.1.1. Métodos Jerárquicos

Se crea un conjunto de grupos anidados, organizados como un árbol jerárquico. Tienen como objetivo realizar agrupaciones sucesivas o divisiones, que minimicen el error o maximice la similitud entre los ejemplos.

Los métodos jerárquicos están divididos en dos grupos (Tan P., et al. 2005):

- Los métodos aglomerativos:

  Comienzan con la creación de un grupo para cada uno de los ejemplos individuales, seguidamente se van mezclando en pares los grupos más cercanos hasta que queda uno (o K grupos).

- Los métodos divisivos:

  Constituyen el proceso inverso a los aglomerativos; comienza con la creación de un grupo que contienen todos los ejemplos y, a partir de ese grupo inicial, se va dividiendo hasta que cada grupo contiene un solo ejemplo (o se crearon K grupos).

Estos métodos necesitan la definición a priori de la cantidad de clusters (K) que se desean formar.

A continuación se presenta el funcionamiento de uno de los métodos, ya que ambos siguen los mismos principios, solo cambian la forma de la creación de los grupos.

Algoritmo de agrupamiento aglomerativo:

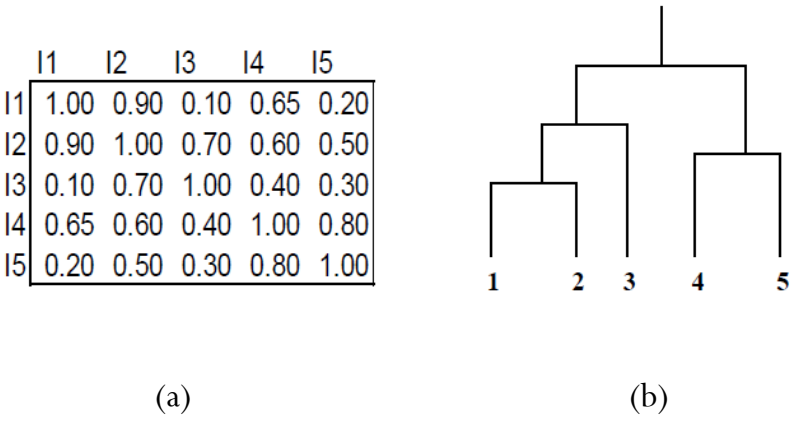
Este método es uno de los más comúnes para la creación de grupos, y se basa en la construcción de una matriz de proximidad, en la cual se ubican

en cada casilla la distancia de separación o similaridad entre grupos. El algoritmo es el siguiente:

1. Calcular la matriz de proximidad.
2. Crear un grupo por cada ejemplo.
3. Repetir
    - Combinar los dos grupos más cercanos
    - Actualizar la matriz de proximidad

Hasta que solo se obtenga un grupo (o K grupos).

La matriz de proximidad viene dada por la distancia entre los datos de entrenamiento, como se muestra en la Figura 2.8-(a). En la Figura 2.8-(b), se dibuja el árbol jerárquico que representa este ejemplo, en el cual, se ordenan los pares de grupos con la proximidad más pequeña.



	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00

(a) (b)

Figura 2.8. (a) Matriz de proximidad entre 5 ejemplos (b) Árbol jerarquico construido para los 5 jemplos según su distancia.

El punto clave de este algoritmo es la construcción de la matriz de proximidad, para así, detectar los dos grupos más cercanos y realizar la mezcla. A continuación se presenta un breve resumen de las métricas que permiten calcular la proximidad entre dos grupos:

i. Distancia mínima o similitud máxima

La distancia[3] mínima viene dada por la distancia más pequeña entre dos ejemplos de dos grupos, o la similaridad[4] máxima dada en la matriz de proximidad. En la Figura 2.9-(a), se observa gráficamente los ejemplos que cumplen con la distancia mínima o similitud máxima entre dos grupos.

ii. Distancia Máxima o similitud mínima

La distancia máxima viene dada por la distancia más grande entre ejemplos de dos grupos, o la similaridad máxima dada en la matriz de proximidad (véase la Figura 2.9-(b)).

iii. Distancia Promedio

Es el promedio de las distancias entre todos los ejemplos de un grupo contra todos los ejemplos del otro grupo a comparar (véase la Figura 2.9-(c)).

iv. Distancia al centroide

Se ubica el centroide de ambos grupos, y se calcula la distancia entre ellos. Se determina que los grupos son vecinos si sus distancias están a una mínima distancia (umbral), como se muestra en la Figura 2.9-(d).

[3]distancia diferencia: desemejanza notable entre un objeto y otro.
[4]similaridad: parecido o semejanza que hay entre dos o más objetos debido a sus características.

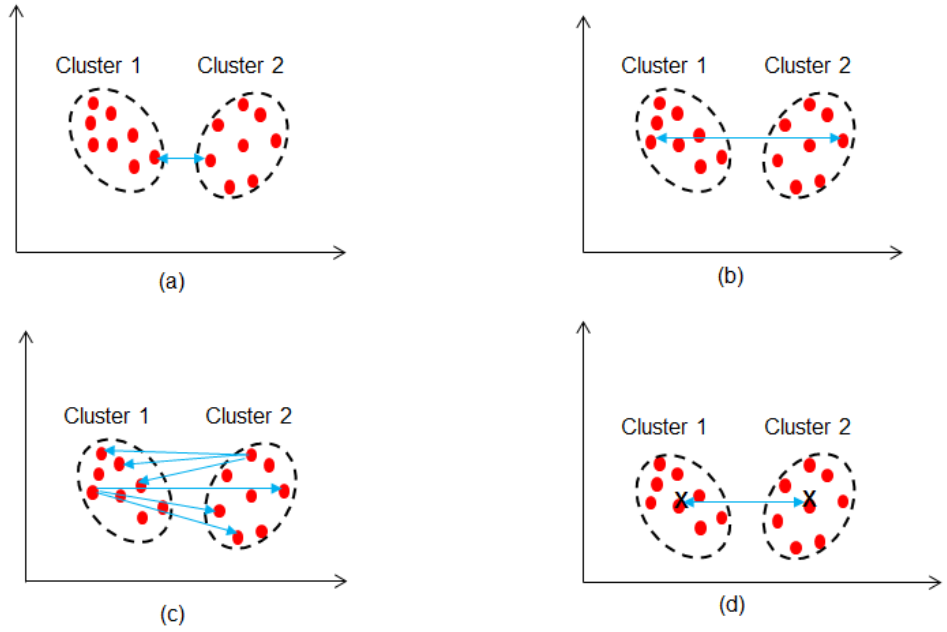


Figura 2.9. (a) Distancia mínima (b) Distancia máxima (c) Distancia de promedio del grupo (d) Distancia con respecto al centroide.

En la figura 2.10 se observa el resultado del agrupamiento aglomerativo, al utilizar diferentes métricas para mezclar los grupos. En la imagen los números de color negro son los ejemplos de la base de datos utilizados. Los números en rojo son los grupos que se fueron formando con cada métrica en orden ascendente, siendo 1 el primer grupo que se construyó y 5 el último.

Existen diversas estrategias que pueden ser empleadas para unir los grupos en las diversas etapas o niveles de un procedimiento jerárquico. Ninguno de estos procedimientos proporciona una solución óptima para todos los problemas que se pueden plantear, ya que es posible llegar a distintos resultados según el método elegido, como se muestra en la Figura 2.10. El buen criterio del investigador, el conocimiento del problema planteado y la experiencia, sugerirán el método más adecuado.

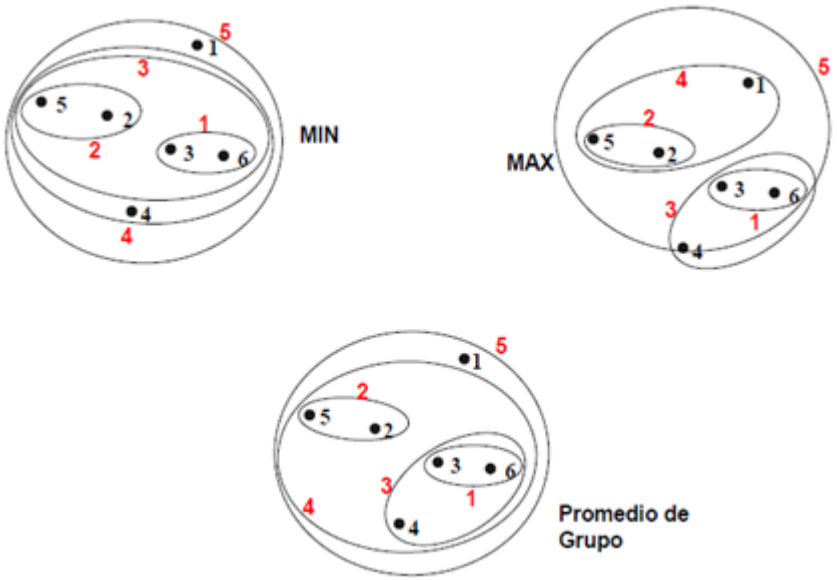


Figura 2.10. Resultados obtenidos al aplicar diferentes metricas de mezcla (tomada de Tan P., et al. 2005).

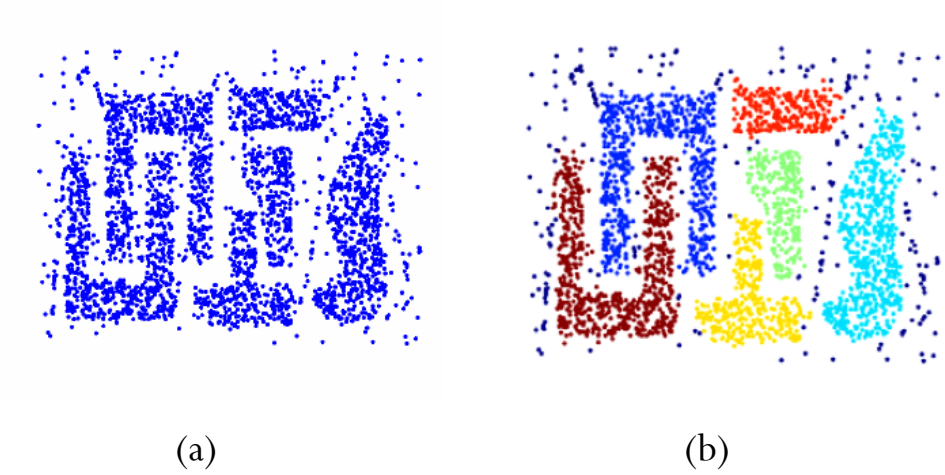
El aglomeramiento jerárquico en general es costoso debido al uso de la matriz de proximidad. El tiempo puede llegar al orden de $O(N^3)$, siendo N el número de ejemplos en el conjunto de entrenamiento. Una vez que se toma una decisión para combinar dos grupos, no puede deshacerse.

2.3.1.2. Algoritmos basados en densidad

Los algoritmos basados en densidad, tratan de formar agrupaciones en áreas con altas densidades de ejemplos, que además, se encuentran rodeados por bajas densidades de ejemplos.

Dichos algoritmos son óptimos cuando la base de datos cuenta con mucho ruido, ya que detecta dichas anomalías y las separa de los grupos con altas densidades. El algoritmo más conocido es DBSCAN (Tan P., et al. 2005), el mismo comienza eliminando los puntos de ruido, es decir que no tienen una densidad mayor a un umbral en un radio previamente definido, seguidamente se procede a realizar el agrupamiento de los puntos restantes. En la Figura 2.11-(a) se muestran los datos originales,

y en la Figura 2.11-(b) se puede observar el resultado obtenido con este algoritmo, los puntos marcados con azul oscuro representan ruido para el algoritmo y son eliminados.



(a) (b)

Figura 2.11. (a) Datos originales (b) Datos despues de evaluarlos en DBSCAN(tomado de Tan P., et al. 2005).

Utilizando el enfoque de densidad, DBSCAN clasifica un punto (ejemplo) como: un punto núcleo que se encuentra en el interior de una zona densa, un punto frontera que se encuentra en el borde de una region densa, y un punto de ruido que se encuentra en una region escasamente ocupada.

DBSCAN requiere dos parámetros: *Eps*, que es el radio de covertura para la búsqueda de puntos; y *minPtos*; que es el mínimo número de puntos requeridos para formar una región densa. Un ejemplo es un punto núcleo si el número de puntos, en un radio de tamaño *Eps,* es mayor a *minPtos.* Un ejemplo es un punto frontera si no es un punto núcleo, y se encuentra dentro de la región de un punto núcleo (esa región también es otro parámetro del algoritmo). Por último, un punto de ruido es aquel que no fue etiquetado como punto núcleo o punto frontera. En la Figura 2.12 se observa que *A* es un punto nucleo debido a que contiene más de 5 puntos en un radio *Eps, B* es un punto frontera ya

que no es un punto núcleo y pertenece a la zona de *A*. Por otro lado, *C* no es ni un punto núcleo ni un punto frontera, por lo tanto se etiqueta como un punto de ruido.

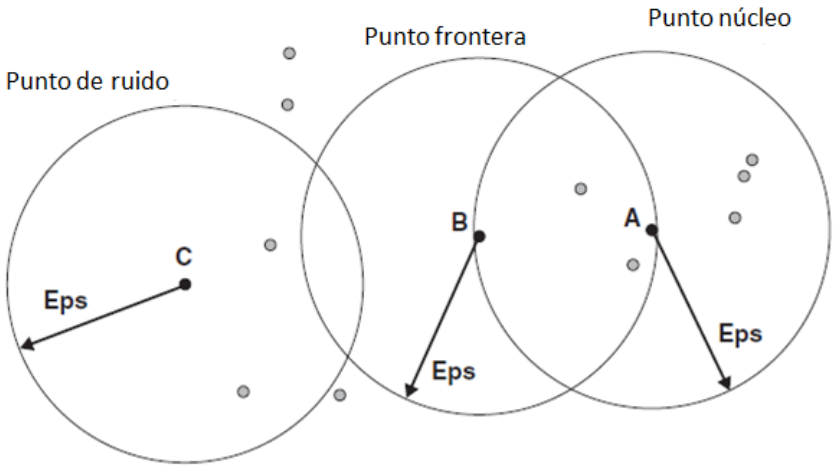


Figura 2.12. Ejemplo de la clasificacion de los ejemplos por DBSCAN (tomado de Tan P., et al. 2005).

De manera general, los pasos que se realizan para formar los grupos con DBSCAN son:

1. Etiquetar todos los ejemplos como puntos núcleo, puntos frontera y puntos de ruido, según corresponda.
2. Eliminar los puntos de ruido.
3. Unir todos los puntos núcleos conectados y convertirlos en un grupo
4. Asignar cada punto frontera al grupo del punto núcleo asociado, según si pertenece a su región.

El tiempo de complejidad de este algoritmo, viene dado por la evaluación de cada uno de los ejemplos de la base de datos y el tiempo que se tarda en encontrar los vecinos dentro del radio *Eps* de cada uno de ellos. En el peor de los casos, la complejidad es del orden $O(N^2)$, siendo N la cantidad de ejemplos. Normalmente, este tipo de algoritmos utiliza indexación para acelerar el proceso y disminuir la complejidad.

2.3.1.3. Algoritmos basados en prototipos

Un grupo se define por un conjunto de objetos, donde cada objeto está más cerca (o es más similar) al prototipo que define al grupo donde fue asignado, que a cualquier otro grupo existente. El prototipo que define al grupo normalmente denota sus características principales; por ejemplo, para atributos solo numéricos, el prototipo del grupo a menudo se representa como el centroide. Si los atributos son nominales, normalmente se utiliza el medoid[5], es decir, el punto más representativo del grupo (Tan P., et al. 2005).

Un algoritmo basado en prototipos es K-means, el cual va particionando el espacio donde se encuentran distribuidos los datos, hasta conseguir *K* grupos que minimicen el error cuadrático de la distancia entre los ejemplos pertenecientes a cada grupo y su centroide correspondiente. Es necesario proveer al algoritmo el valor de *K* para obtener el resultado.

En la Figura 2.13 se muestra el algoritmo de K-means, las entradas son el número de grupo a crear *K* y el conjunto de entrenamiento. El proceso comienza inicializando *K* centroides. Una vez realizado esto, se asignan los puntos más cercanos a los *K* centroides y se vuelve a calcular los centroides con dichos puntos, este proceso se repite hasta que haya una convergencia del error cuadrático de la distancia.

[5]medoid es el ejemplo con la disimilitud promedio minima (o similaridad máxima), con respecto los demás ejemplos pertenecientes a su grupo

Input:

- K (number of clusters)
- Training set $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

Repeat {

for i =1 to m

$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$

for k = 1 to K

μ_k := average (mean) of points assigned to cluster

} until convergence criteria is met

Figura 2.13. Macro algoritmo de K-means.

En la Figura 2.14 se muestra una demostración gráfica del proceso que realiza K-means con *K* igual a 2. Inicialmente se determina aleatoriamente dos centroides, en la primera iteración se asignan los ejemplos más cercanos a cada uno de los centroides y se crean dos grupos. Seguidamente se calcula el nuevo centroide con los ejemplos de cada grupo. Es en la segunda iteración se observa el movimiento de los centroides. En las siguientes iteraciones, el centroide se va moviendo hasta que consigue una configuración en la que el error converge a un valor mínimo.

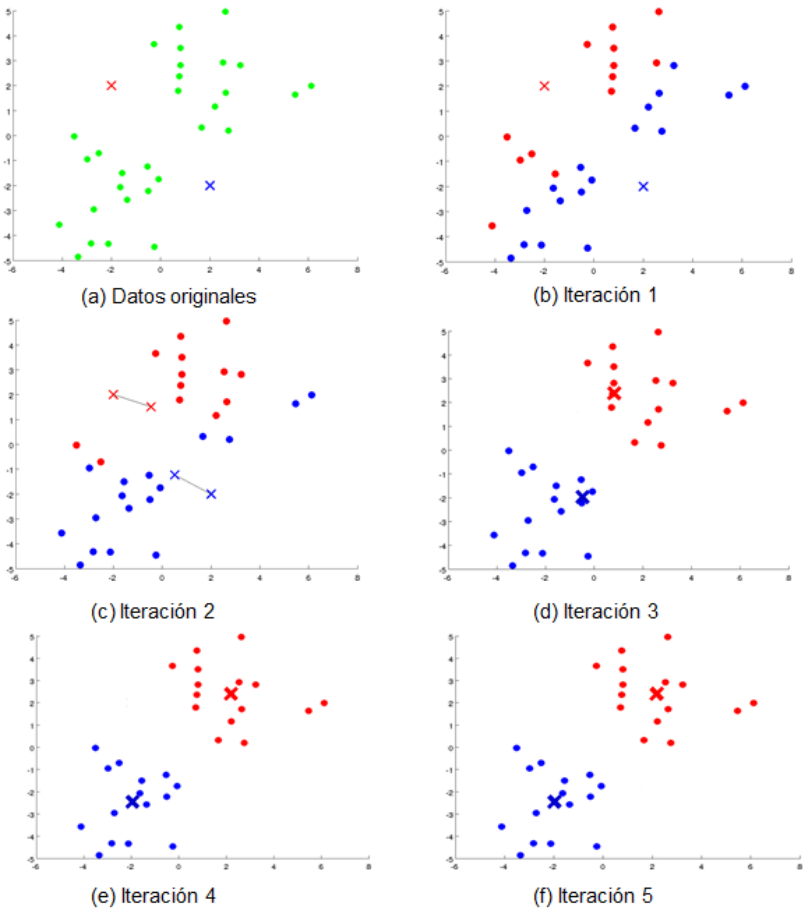


Figura 2.14. Corrida grafica de K-means con K igual a 2.

2.3.1.4. Algoritmos basados en teoría de grafos

Desde el punto de vista de la MD, cada instancia se puede representar como un nodo y los arcos como las conexiones entre las instancias, de esta manera se puede utilizar la teoría de grafos para definir el cluster como un componente fuertemente conectado. Los arcos pueden representar la distancia entre los objetos, u otras caracteristicas representativas de similaridad o relación entre nodos (Hastie T. et al, 2009). Sin embargo, desde el punto de vista de la teoría de grafos, existen diversas aproximaciones para la creación de grupos que estan relacionados con la construcción y partición de grafos (Takashi W. and Hiroshi M. ,2003). Este problema es NP-Completo, y consiste en

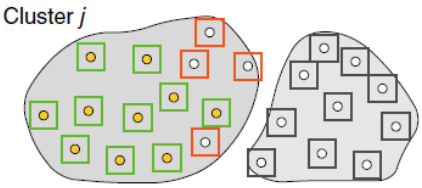
encontrar una partición de los nodos tal que se minimize un costo, usualmente, el de la suma de los arcos entre diferentes particiones.

2.3.2. Métricas para evaluar un algoritmo de agrupamiento

Previamente se dieron las métricas que permiten evaluar los modelos de clasificación, en ese caso era posible la verificación de los resultados obtenidos debido a que se cuenta con las etiquetas de cada ejemplo del conjunto de prueba. Ahora bien, se discutirá cómo realizar la evaluación de los grupos como resultado de los distintos algoritmos estudiados. Sabemos que los ejemplos no cuentan con etiquetas, y solo se estaría confiando en que la agrupación realizada es correcta. Pero eso no es suficiente, por eso son necesarias ciertas medidas.

Las medidas numéricas de validación son clasificadas en tres tipos (Tan P., et al. 2005):

- Índice Externo: usado para medir el grado en que las etiquetas de un grupo coinciden con etiquetas de clases externas. Esta prueba se basa en el conocimiento a priori de las etiquetas de los grupos, es decir, se sabe a cuál grupo pertenece cada ejemplo. A continuación se presentan dos ejemplos de estas métricas:
  i. F-mesure: esta métrica está basada en la precisión y recall, al igual que en el caso de clasificación. Una vez entrenado el algoritmo de agrupamiento, se procede a observar las agrupaciones que se formaron, y con las etiquetas originales de los datos se puede crear la matriz de confusión como se muestra en la Figura 2.15. Siendo TP los verdaderos positivos, FP falsos positivos, FN falsos negativos y TN los verdaderos negativos.



		Truth	
		P	N
Hypothesis	P	TP	FP
	N	FN	TN

Figura 2.15. Formación de la matriz de confusión con dos grupos.

Dada la Figura 2.15, las ecuaciones en el ítem 2.2.2 se mantienen para el cálculo de la precisión, recall y F-score.

ii. Entropía: Es el grado de coincidencia de los grupos a las clases ya definidas de los datos originales. Para cada grupo se calcula la distribución de los datos (Tan P., et al. 2005). Es decir, para cada grupo se calcula la probabilidad de que cada objeto en el grupo i pertenezca a la clase j, con la siguiente ecuación:

$$p_{ij} = \frac{m_{ij}}{l_i} \qquad (2.9)$$

Donde, l_i es el número de objetos en el grupo i y m_{ij} es el número de objetos de la clase j en el grupo i. Por lo tanto, la entropía de cada grupo se puede calcular con la ecuación clásica dada en la ec.(2.10), siendo C el número de clases.

$$e_i = -\sum_{j=1}^{C} p_{ij} log_2 p_{ij} \qquad (2.10)$$

Para el cálculo de la entropía del conjunto completo del grupo, se calcula la suma de las entropías de cada uno

con un peso asociado, como está dado en la ec. (2.11), siendo *K* el número total de grupos y *l* el número total de ejemplos. Una entropía pequeña implica que las agrupaciones realizadas son correctas, caso contrario ocurre cuando la entropía es grande.

$$e = \sum_{i=1}^{K} \frac{l_i}{l} e_i \qquad (2.11)$$

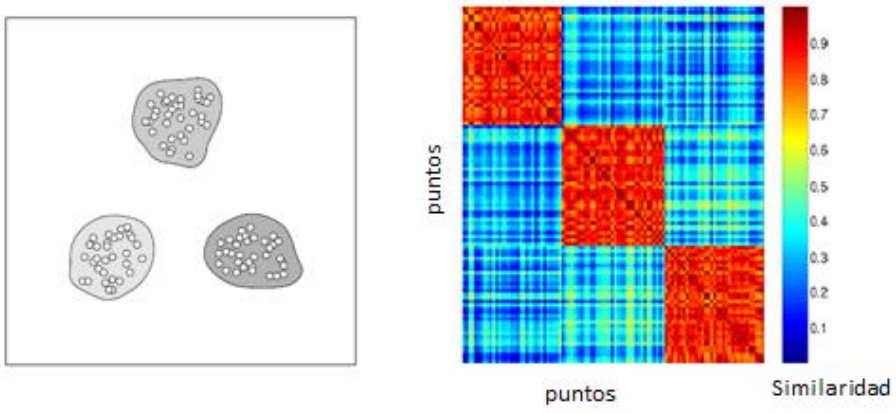
- Índice Interno: usado para medir la "bondad" de un estructura agrupada sin tener información de referencia externa (no se conocen las etiquetas de los ejemplos). Existen diferentes métricas pertenecientes a esta clasificación, como las siguientes:

  i. Coeficiente de correlación: dada la matriz de similaridad del conjunto de ejemplos (que se obtiene mediante el cálculo de la similaridad entre todos los ejemplos), y una matriz de similaridad "correcta" obtenida por un algoritmo cualquiera de agrupamiento. La matriz de similaridad "correcta" se construye creando una matriz que tiene una fila y una columna por cada ejemplo, y asignando similaridad 1 cuando un par de ejemplos pertenece al mismo grupo y 0 en otro caso. Se puede evaluar la bondad del algoritmo calculando la correlación entre la matriz de similaridad de los ejemplos y la versión "correcta" de la matriz de similaridad.

     Otra forma de evaluar el resultado del algoritmo es graficando la matriz de similaridad de los ejemplos y los grupos dados por el algoritmo, para así, juzgar visualmente sobre el resultado obtenido. Para graficar

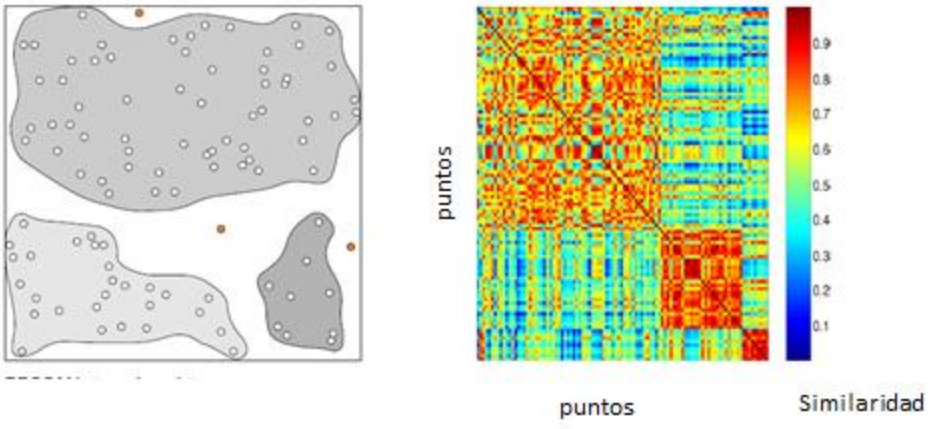
la matriz de similaridad se ordenan los puntos con respecto a las etiquetas de cada grupo. Es decir, si se obtuvieron dos grupos, los puntos del primer grupo se ubican en las primeras filas y columnas de la matriz, mientras que los ejemplos del segundo grupo se ubican al final. Finalmente se calcula la similaridad entre todos los ejemplos ya ordenados y se obtiene la gráfica.

En la Figura 2.16(a) se puede observar el resultado obtenido por K-means para una base de datos aleatoria, en el cual se obtuvo tres grupos bien definidos. Si se grafica la matriz de similaridad ordenada por las etiquetas obtenidas por K-means, se obtiene el resultado de la Figura 2.16-(b), en dicha figura se puede observar tres áreas bien definidas con tonos rojos, mientras más roja sea la tonalidad la similaridad tiende a 1, mientras más baja la similaridad tiende a un color azul oscuro. Caso contrario ocurre en la Figura 2.17, para los resultados obtenidos por DBSCAN, se diferencian tres áreas de una similaridad moderadamente altas, pero no se puede concluir mucho con estos resultados.



(a) (b)

Figura 2.16. (a) Tres grupos bien separados y resultado obtenido con K-means. (b) Matriz de similaridad entre los puntos ordenada por las etiquetas obtenidas por K-means(tomado de Tan P., et al. 2005).



(a) (b)

Figura 2.17. (a) Cluster con datos aleatorios resultado obtenido con DBSCAN. (b) Matriz de similaridad ordenada por las etiquetas obtenidas por DBSCAN (tomado de Tan P., et al. 2005).

ii. Cohesión y separación

Está asociado a la distancia entre los grupos y a la dispersión de los objetos dentro del mismo grupo.

En la Figura 2.18 se denota gráficamente la cohesión y la separación de los clusters. La cohesión desde el punto de vista de los algoritmos basados en prototipos, puede

ser vista como la proximidad de cada ejemplo a su centroide (o medoid), mientras que la separación entre dos grupos puede ser medida como la proximidad entre ellos. Se pueden encontrar diferentes variaciones del cálculo de cohesión y separación, por ejemplo la separación se puede medir con la diferencia entre los centroides de los grupos.

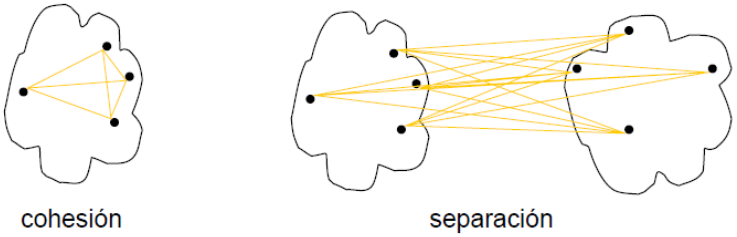


Figura 2.18. Cohesión y separación de clusters.

- Índice Relativo: usado para comparar dos agrupamientos diferentes. Frecuentemente se usa una combinación del índice externo e interno para esta métrica. Por ejemplo, dos grupos creados por K-means pueden ser comparados ya sea usando el error cuadrático de la distancia o la entropía.

2.4. Algoritmos semi-supervisados

Toman las ventajas más resaltantes del aprendizaje supervisado y no supervisado. Es utilizado solamente para mejorar los resultados de la clasificación. Lo que se hace es dividir el conjunto de entrenamiento en dos subconjuntos, en los cuales uno de ellos contará con sus etiquetas mientras que el otro no. Seguidamente se entrena el clasificador de una manera

diferente denominada co-entrenamiento (co-training) (Chapelle, O et al, 2006).

La idea de co-trainnig es, dado un problema de clasificación, se dividen los ejemplos de la base de datos en dos conjuntos. El primer conjunto de datos etiquetado se utiliza para entrenar un modelo de clasificación, mientras que al segundo se le remueve la etiqueta. Los ejemplos del segundo conjunto, a los cuales se les removió la etiqueta, son etiquetados por medio de su evaluación en el modelo obtenido con los datos etiquetados. El modelo final resultante viene dado una vez que se culminan las dos fases anteriores.

Existen diferentes suposiciones que pueden validar el uso de algoritmos semi-supervisados en lugar de los algoritmos supervisados, como se definen a continuación (Chapelle, O et al, 2006):

- Suposición de suavidad

  Si dos ejemplos de la base de datos *x1* y *x2* en una región de alta densidad están cerca, entonces debería haber correspondencia entre su salidas *y1* y *y2*. Esta suposición implica que si dos puntos están unidos por un camino de alta densidad (por ejemplo, si pertenecen al mismo grupo), es probable que sus salidas están cerca. Si, por otra parte, están separados por una región de baja densidad, sus salidas necesariamente están cerca.
- Suposición de cluster

  Si dos puntos están en el mismo cluster, es probable que ambos pertenezcan a la misma clase.
- Suposición de manifold[6]

  El problema de dimensionalidad afecta a muchos algoritmos de MD, disminuyendo su desempeño a medida que la dimensión de los ejemplos aumenta (Verleysen M. y Damien François, 2005). Una base de datos conformada por ejemplos de dimensiones elevadas, puede reducirse en

[6]manifold: un espacio matemático abstracto semejante a los espacios descritos por la geometría euclídeana

un manifold de baja dimensión. Con esta suposición se intenta realizar aprendizaje sobre el manifold, para evitar el problema de la dimensionalidad, utilizando los algoritmos semi-supervisados.

Chapelle O y sus colaboradores (2006), presentan en su revisión una gran variedad de algoritmos semi-supervisados, los mismos utilizan solo el enfoque supervisado o el enfoque supervisado, combinado con el enfoque no supervisado, para resolver problemas semi-supervisados. Entre los más resaltantes se encuentran: 1-NN, Cluster-Kernel, MVU+1-NN, LEM+1-NN, Discrete Regression, entre otros.

2.5. Principal Component Analysis (PCA)

Es uno de los algoritmos más conocidos y utilizados para la reducción de dimensión, así como también, para la visualización de los datos. Esta herramienta será ampliamente utilizada en nuestra propuesta para visualizar los datos y para comprender mejor su comportamiento.

2.5.1. Formulación del problema de PCA

Sin perder generalidad del problema, supóngase que se tiene una base de datos en $\mathbb{R}^2$ y se desea reducir de dos dimensiones (2D) a una dimensión (1D). Los datos están distribuidos en el plano bidimensional como se muestra en la Figura 2.20.

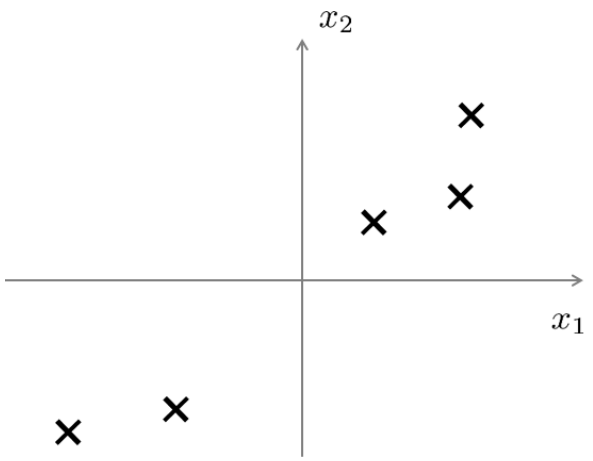


Figura 2.19. Datos en el plano bidimensional.

La reducción de este ejemplo de dos dimensiones a una dimensión, puede traducirse a buscar una recta donde se puedan proyectar los datos de la Figura 2.19, la cual minimice la distancia entre el punto original y la proyección sobre la recta, como se muestra en la Figura 2.20.

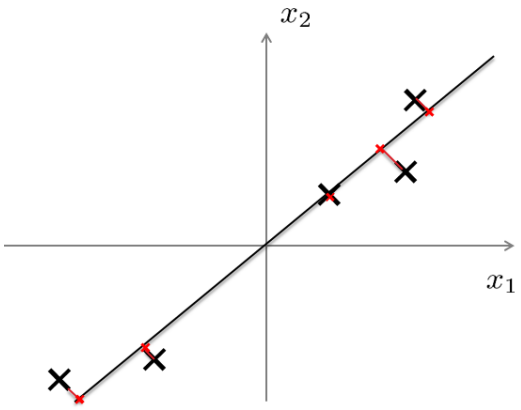


Figura 2.20. Proyección de los datos en 2D a 1D sobre una recta.

El objetivo principal de PCA es encontrar una superficie con una dimensión menor (en nuestro ejemplo la recta) en la cual se pueda proyectar los datos que satisfagan que la suma de los errores cuadráticos de la distancia (comúnmente llamado error de proyección en PCA), entre el punto original y el punto proyectado, sea minimizada.

Para reducir de dos dimensiones a una dimensión, se debe buscar un vector director $u^{(1)} \in \mathbb{R}^2$ para proyectar los datos, el cual minimice el error de proyección. Dicho vector da la dirección de la recta de la Figura 2.20.

Para reducir de n dimensiones a k dimensiones, se deben buscar k vectores $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ para proyectar la data, que minimice el error de proyección. En la Figura 2.21 se puede observar un ejemplo de este caso, el cual trata de proyectar datos en tres dimensiones (3D) a dos dimensiones (2D). Se calculan dos vectores con dirección $u^{(1)}$ y $u^{(2)}$, que definen el plano que minimiza el error de proyección.

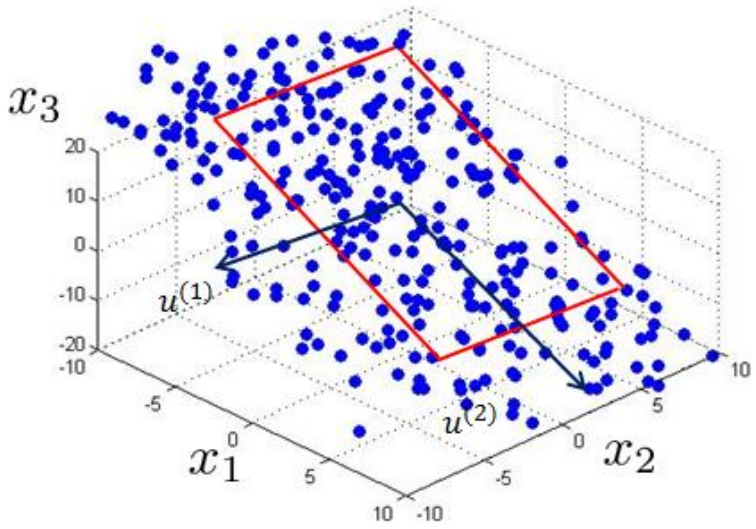


Figura 2.21. Proyección de los datos en 3D a 2D sobre un plano.

2.5.2. Algoritmo de PCA

PCA se define matemáticamente como una transformación lineal ortogonal que transforma un conjunto de datos $x \in \mathbb{R}^n$ a un nuevo sistema de coordenadas $z \in \mathbb{R}^k$, en el cual $k < n$. Para reducir vectores n-dimensionales a k-dimensionales, se deben encontrar k vectores $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ en los cuales proyectar los datos, que a su vez, minimicen el error de proyección (Jolliffe I., 2002).

El procedimiento para determinar los vectores *u,* parte del cálculo de la matriz de covarianza sigma de los datos (ec. 2.12):

$$Sigma = \frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)}\right)\left(x^{(i)}\right)^T \quad (2.12)$$

Seguidamente se determina los autovectores de Sigma. Existen diversas funciones en paquetes matemáticos que permiten realizar el cálculo de los autovectores de la matriz de covarianza, como "svd" o

"eig" de octave[7], dichas funciones retornan una matriz *U* de dimensión *nxn*. Cada columna de la matriz *U* es un vector (con una dirección), y viene dado por la siguiente matriz:

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

Para realizar una reducción de *n* dimensiones a *k* dimensiones, se deben tomar los primeros *k* vectores de la matriz de *U,* esta matriz reducida se denomina U_r y es de dimensión *nxk*.

Finalmente, para obtener el nuevo sistema de coordenadas se utiliza la ec.(2.13). Dado que la entrada $x^{(i)}$ es de dimensión *nx1* y U_r es de dimensión *nxk* , $z^{(i)}$ queda como un vector de dimensión *kx1,* logrando así el objetivo propuesto.

$$z^{(i)} = U_r^T * x^{(i)} \quad (2.13)$$

Estadísticamente, esta reducción es válida debido a que se agrupan los atributos de la bases de datos que están altamente correlacionados, y es comúnmente utilizado para el procesamiento de datos antes de utilizar cualquier algoritmo de MD. PCA puede incurrir en pérdida de información, cuando se reducen las dimensiones más de lo debido, es por ello, que se debe ser cuidadoso al seleccionar *k*. Sin embargo, ha demostrado ser útil para la visualización de datos con altas dimensiones.

Las bases teóricas expuestas, permitieron tener un amplio conocimiento del contexto en el cual se basa la presente investigación, además, se resaltaron teorías que serán ampliamente utilizadas para el diseño de nuestra propuesta.

[7]octave https://www.gnu.org/software/**octave**/

Capítulo 3

Diseño del algoritmo

En este capítulo se presenta el diseño teórico del algoritmo híbrido para realizar clasificación y agrupamiento, así como también, las hipótesis en las que el algoritmo se fundamenta.

3.1. Características del algoritmo

El principal objetivo de esta propuesta es que, en un solo sistema, se puedan realizar tareas de clasificación o de agrupamiento, además, de un caso especial que lo denominaremos híbrido. El caso híbrido surge cuando en una base de datos se requiera hacer ambas tareas en conjunto (clasificación y agrupamiento), y será explicado con detalle más adelante. Cabe destacar, que la presente propuesta está diseñada para resolver solo problemas con bases de datos que cuentan con todos sus atributos numéricos.

A continuación se presentan las características funcionales más importantes con las que cuenta el algoritmo propuesto:

1. Dada una base de datos sobre la cual se requiere realizar MD con ejemplos etiquetados, realizar el entrenamiento del modelo para generar prototipos (denominados clases).

2. Dada una base de datos sobre la cual se requiere realizar MD con ejemplos no etiquetados, realizar el entrenamiento del modelo y generar prototipos (denominados grupos).
3. Dada una base de datos sobre la cual se requiere realizar MD con ejemplos no etiquetados y etiquetados, el sistema debe ser capaz de construir prototipos (grupos o clases) siguiendo una serie de criterios.
4. Dado el sistema ya entrenado, ser capaz de realizar clasificación o agrupamiento de nuevos ejemplos.
5. Dado el sistema ya entrenado, ser capaz de crear nuevos prototipos.

3.2. Macro algoritmo general

El algoritmo está conformado por dos etapas principales, que son habituales en los algoritmos de MD de clasificación, como lo son la de entrenamiento y de funcionamiento. De manera general, el algoritmo sigue las actividades presentadas en la Figura 3.1.

La primera actividad dada en la Figura 3.1 es realizar la lectura de los datos de la base de datos proporcionada, dicha base de datos viene organizada en una tabla estándar para problemas de MD, denominada *Vista Minable* (ver sección 2.1.3). Los archivos en donde se almacena esta tabla pueden estar en formato .arff, .txt, xls, entre otros. Esta sección permite la lectura de los archivos para procesar la *Vista Minable,* y adaptarla a la estructura de nuestro diseño. Seguidamente se realiza el proceso de entrenamiento del modelo con dicha base de datos. En esta etapa se crean los prototipos (clases o grupos) del modelo de MD. Una vez generados los prototipos, se procede a validarlos con métricas clásicas de clasificación y/o agrupamiento.

La fase de funcionamiento se inicia si y solo si los prototipos fueron validados, y hay entradas para clasificar/agrupar (Véase la Figura 3.1). En el caso particular de

clasificación para validar los prototipos, normalmente se utiliza un conjunto extraído de la base de datos que se denomina datos de prueba, y este conjunto es evaluado en el funcionamiento del algoritmo. En la fase de funcionamiento se genera como salida la clase/grupo a la cual pertenece una entrada, y este proceso es recursivo hasta que se hayan evaluado todas las entradas, como se muestra en la Figura 3.1.

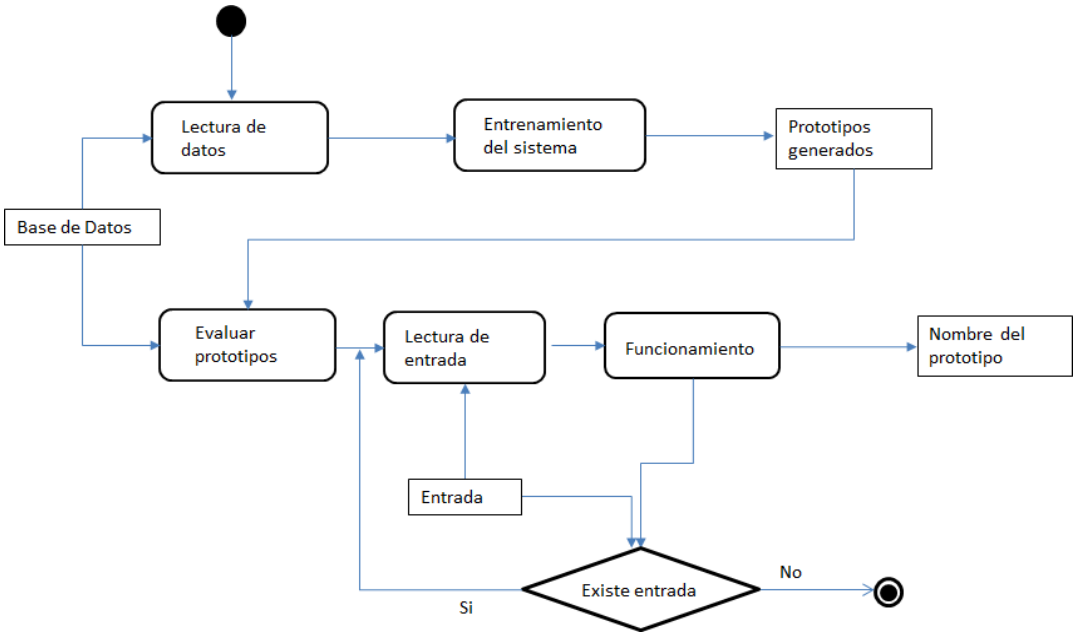


Figura 3.1. Diagrama de actividades del macro algoritmo general.

Una vez definidas las actividades que va a realizar nuestra propuesta, se presenta su Macro Algoritmo (MA) 3.1.

Macro Algoritmo 3.1. Propuesta general

- **Entradas:** Dirección física del archivo que contiene la base de datos
- **Procedimiento:**
  1. Realizar lectura de la base datos
  2. Entrenar el sistema
  3. Evaluar los prototipos
  4. Funcionamiento del sistema
  5. Fin
- **Salida:** Prototipos *P* generados por el algoritmo

A continuación se presentan cada uno de los componentes del Macro Algoritmo 3.1.

3.2.1. Lectura de la base de datos (MA 3.1. ítem 1)

Normalmente las bases de datos destinadas para MD cuentan con diferentes formatos. Esta etapa tiene como finalidad realizar la lectura de la base de datos y adaptarla al formato válido para el sistema, el cual será especificado en el capítulo de implementación del algoritmo. Sus pasos más resaltantes se muestran en el Macro Algoritmo 3.2.

Macro Algoritmo 3.2. Lectura de base de datos

- **Entradas:** Dirección física del archivo que contiene la base de datos
- **Procedimiento:**
  1. Lectura del archivo
  2. Verificación del formato del archivo
  3. Lectura de las instancias
  4. Transformación de las instancias al formato válido para el sistema
  5. Fin
- **Salida:** Arreglo de instancias *I* con los atributos y sus etiquetas (esto último, cuando sea el caso)

3.2.2. Etapa de entrenamiento (MA 3.1. ítem 2)

3.2.2.1. Modelo teórico

El algoritmo diseñado se fundamenta en la construcción de grupos basados en prototipos (sección 2.3.1.3), utilizando el concepto de similaridad con el centroide y la estrategia del vecino más cercano 1-NN.

Se dice que un ejemplo es similar a una clase/grupo, si dicho ejemplo es semejante al centroide de la clase/grupo. En este sentido, se pueden crear clases y grupos reuniendo los ejemplos con características semejantes. Por lo tanto, en esta propuesta una instancia se va a asignar provisionalmente a la clase/grupo con la

similaridad más alta, esto quiere decir que dicha asignación se puede modificar en la fase de entrenamiento, bajo ciertas condiciones que presenta el algoritmo que serán detalladas a lo largo de la propuesta.

Por otro lado, el vecino más cercano a un ejemplo es el que tiene la distancia más corta o la similaridad más alta. La estrategia del vecino más cercano se utiliza para fortalecer la asignación por similaridad al centroide, es aquí donde la asignación provisional generada por la similaridad al centroide puede cambiar.

Para comprender la función del vecino más cercano, se presenta gráficamente el planteamiento (ver figura 3.2). Supóngase que se tiene un grupo creado con cierta cantidad de instancias pertenecientes a él y un centroide que lo representa, seguidamente una entrada nueva es procesada y es asignada sobre dicho grupo usando la similaridad con el centroide. En la Figura 3.2-(a) y (b) se muestran dos posibles nuevas instancias resaltadas en color rojo asignadas al mismo grupo, pero con distancias diferentes. Inicialmente, se realiza una asignación temporal a dicho grupo, se busca el vecino más cercano, y se obtiene la distancia *d* en ambos casos. Si *d* está por debajo de un umbral previamente definido, la instancia pertenece al grupo, como se muestra en la Figura 3.3-(a). Si no, se separan las instancias y se crean dos grupos, como se evidencia en la Figura 3.3-(b).

(a) (b)

Figura 3.2. (a) Asignación provisional de la instancia e_1 y detección del vecino más cercano (b) Asignación provisional de la instancia e_2 y detección del vecino más cercano.

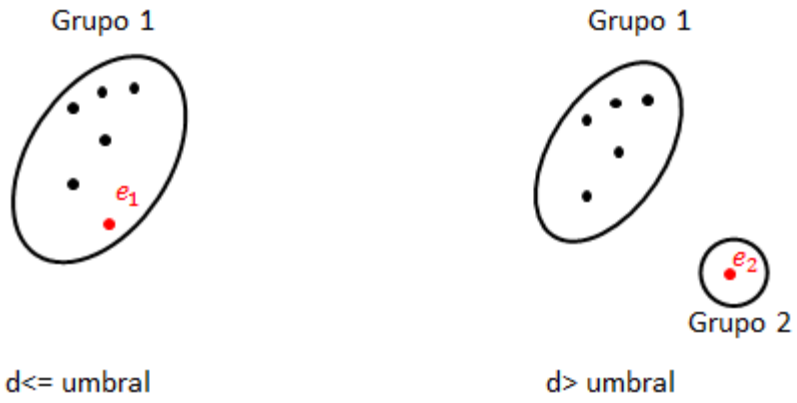


Figura 3.3. (a) Asignación final de la entrada e_1 al grupo 1 (b) Separación de la instancia e_2 y creación de un nuevo grupo.

La definición del umbral de vecindad es de suma importancia, ya que tiene un gran peso para tomar la decisión de asignar un ejemplo a una clase/grupo o realizar la separación de ejemplos.

Inicialmente, el algoritmo necesita un umbral de vecindad para iniciar el proceso de asignación o separación. Se considera que cada clase/grupo tiene su propio umbral de vecindad, que es inicializado con el valor por defecto la primera vez.

A medida que se van evaluando los ejemplos, se va modificando el umbral de vecindad, lo cual básicamente se realiza de manera heurística, calculando la distancia promedio o varianza entre los ejemplos del mismo prototipo.

Cabe destacar que el umbral de vecindad por defecto es introducido por el usuario, y está basado en un estudio a priori de las características de los datos de entrenamiento. Normalmente, la varianza de los datos o el promedio de las distancias entre los ejemplos son buenas maneras para inicializar dicho umbral, debido a que estas métricas generan una buena estimación de la dispersión o distancia entre los ejemplos.

Se definen criterios para la creación, separación, mezcla y modificación de los prototipos, los cuales permiten justificar las decisiones tomadas en dichas operaciones, los cuales son:

i. Si se tiene una instancia de entrada con una etiqueta nueva en el sistema, automáticamente se va a crear una clase nueva con dicha instancia.

ii. Si dos ejemplos fueron asignados al mismo prototipo con alta similaridad y cumple con el umbral de vecindad, ambos ejemplos pertenecen al mismo prototipo.

iii. Si dos puntos (ejemplos) en una región de alta densidad están cerca, entonces debería haber correspondencia entre las salidas de ambos, es decir, los prototipos a los cuales fueron asignados deben ser el mismo.

iv. Si un ejemplo no etiquetado es asignado a una clase con alta similaridad y cumple con el umbral de vecindad, el ejemplo no etiquetado pertenece a la clase asignada y es etiquetado al final del proceso de entrenamiento.

v. En caso de que una nueva instancia asignada por similaridad a un prototipo no cumpla con el umbral de vecindad, y además, con ningún otro prototipo del sistema; se debe crear un nuevo grupo con esta instancia, ya sea etiquetada o no.

vi. Dos prototipos vecinos están separados, ya sea porque existe una distancia grande entre sus fronteras o por baja densidad entre las mismas.

vii. Dos prototipos vecinos deben unirse si existe una distancia muy pequeña entre sus fronteras o por alta densidad entre las mismas.

viii. Se permite la unión de clases vecinas, si y solo si, cumplen con la condición anterior; y además; las clases no se pueden separar linealmente. Es decir, los ejemplos de ambas clases están mezclados en una misma zona.

ix. Se permite la unión de una clase con un grupo, siempre y cuando cumplan la condición vii. El resultado de esta unión va a ser la clase original, con los ejemplos del grupo integrados a ella.

Siguiendo los criterios expuestos, se desea que este planteamiento resuelva tres tareas: Clasificación, Agrupamiento y el caso Híbrido. A continuación, se presentan los principios fundamentales que se seguirán para realizar el entrenamiento de cada tarea:

a. Clasificación:

El proceso de clasificación ocurre cuando se entrena el sistema con ejemplos etiquetados y se crean clases. Este caso será diseñado como un algoritmo supervisado, en el mismo se verifica las etiquetas de los ejemplos para realizar la clasificación.

Para el entrenamiento, el planteamiento general que se va a seguir esta resumido en el diagrama de actividades de la Figura 3.4, en el mismo se muestra el recorrido que sigue una nueva entrada en el proceso de clasificación. Los pasos más resaltantes de dicho diagrama son los siguientes:

- Dada entrada etiquetada, si la etiqueta es nueva en el sistema crear una nueva clase.
- Si no, se ubica la clase más similar al ejemplo con respecto al centroide. Las actividades que se realizan con esta clase (detectada por similaridad con el centroide) están guiadas por las flechas de color rojo en la Figura 3.4, mientras que las actividades que se realizan con clases vecinas a la primera clase están guiadas con flechas de color azul. Una vez ubicada la clase se toma la decisión de:
    - Asignar el ejemplo a la clase cuando cumple con el umbral de vecindad
    - No asignar el ejemplo a la clase si no cumple con el umbral de vecindad, buscar su prototipo vecino más cercano para ser asignado allí, si cumple con el umbral de vecindad.
    - Si no cumple con ninguno de los casos anteriores crear un nuevo grupo. Se considera la creación de un nuevo grupo, debido que a pesar de que la instancia es etiquetada y la etiqueta es conocida por el sistema, la misma no tiene suficientes características semejantes a su propia clase o a otro prototipo existente (se des-etiqueta el ejemplo).

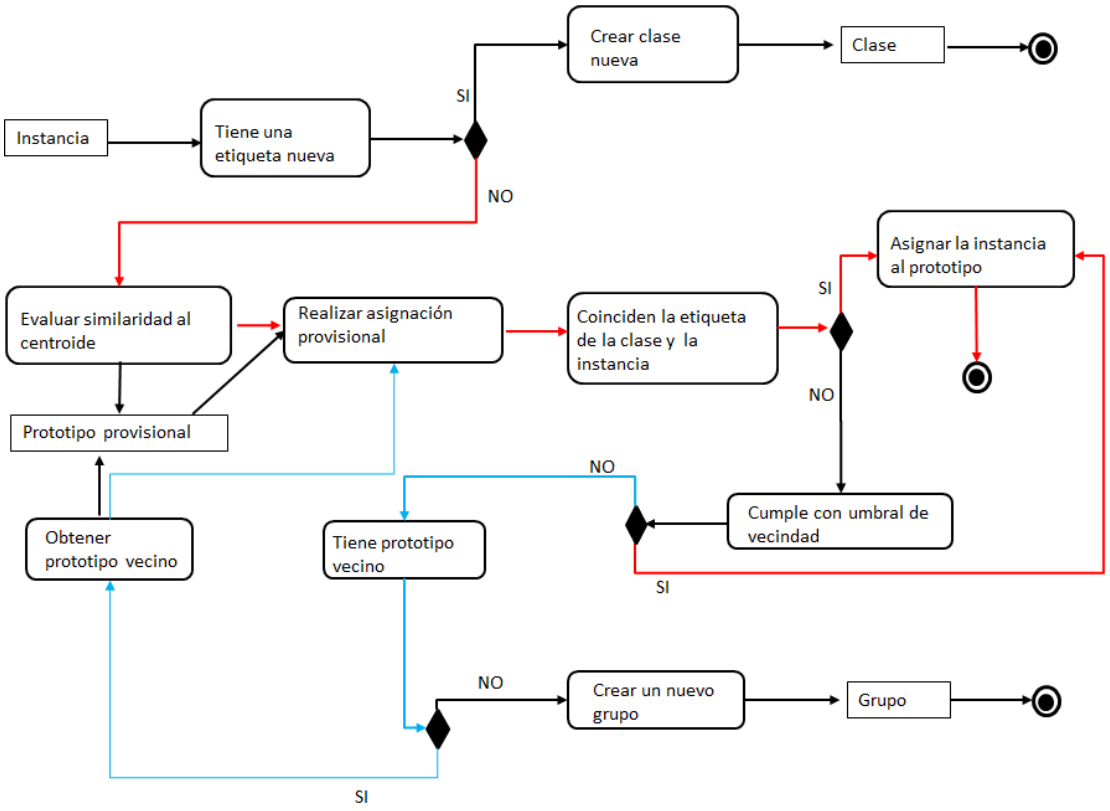


Figura 3.4. Diagrama de actividades de la etapa de clasificación.

En caso de que un nuevo ejemplo en el sistema se desee clasificar una vez culminada la fase de entrenamiento, se asigna dicho ejemplo a la clase con la similaridad más alta y que cumpla con los criterios dados. En la Figura 3.5 se puede observar tres clases bien definidas, donde el punto rojo indica un ejemplo sin etiqueta que fue asignado a la clase 3, por cumplir con los criterios antes mencionados.

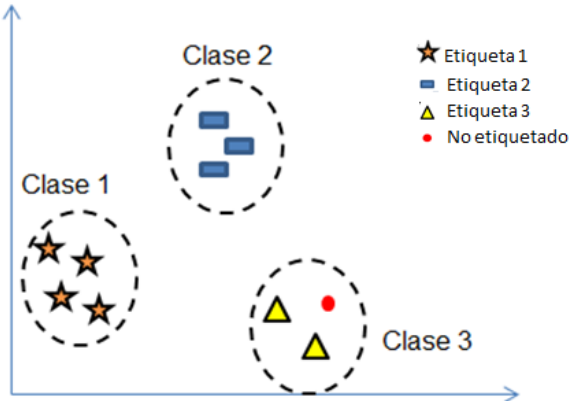


Figura 3.5. Ejemplo de clasificación.

b. Agrupamiento:

El proceso de agrupamiento será trabajado como los algoritmos no supervisados. Al igual que el caso anterior se utilizarán los conceptos de similaridad con el centroide y el vecino más cercano, pero además, se utiliza la métrica de densidad para fortalecer este caso.

Para el entrenamiento del modelo se plantean las actividades dadas en la Figura 3.6. A continuación se presentan sus pasos más resaltantes:

- Ubicar el grupo más similar al ejemplo con respecto al centroide.
- Una vez ubicado el grupo, se ubica su ejemplo vecino más cercano y dadas una serie de condiciones que serán detallas más adelante, asociados a la similaridad y al vecino más cercano, se puede tomar la decisión de:
    - Asignar el ejemplo al grupo, buscar su grupo vecino y verificar condiciones para realizar mezcla entre grupo.
    - No asignar el ejemplo al grupo, y buscar su grupo vecino más cercano para ser asignado allí, si cumple con los criterios. En caso de ser asignado en el grupo vecino, verificar las condiciones para separar instancias del primer grupo para ser asignadas al vecino.
    - Si no cumple con ninguno de los casos anteriores, crear un grupo nuevo

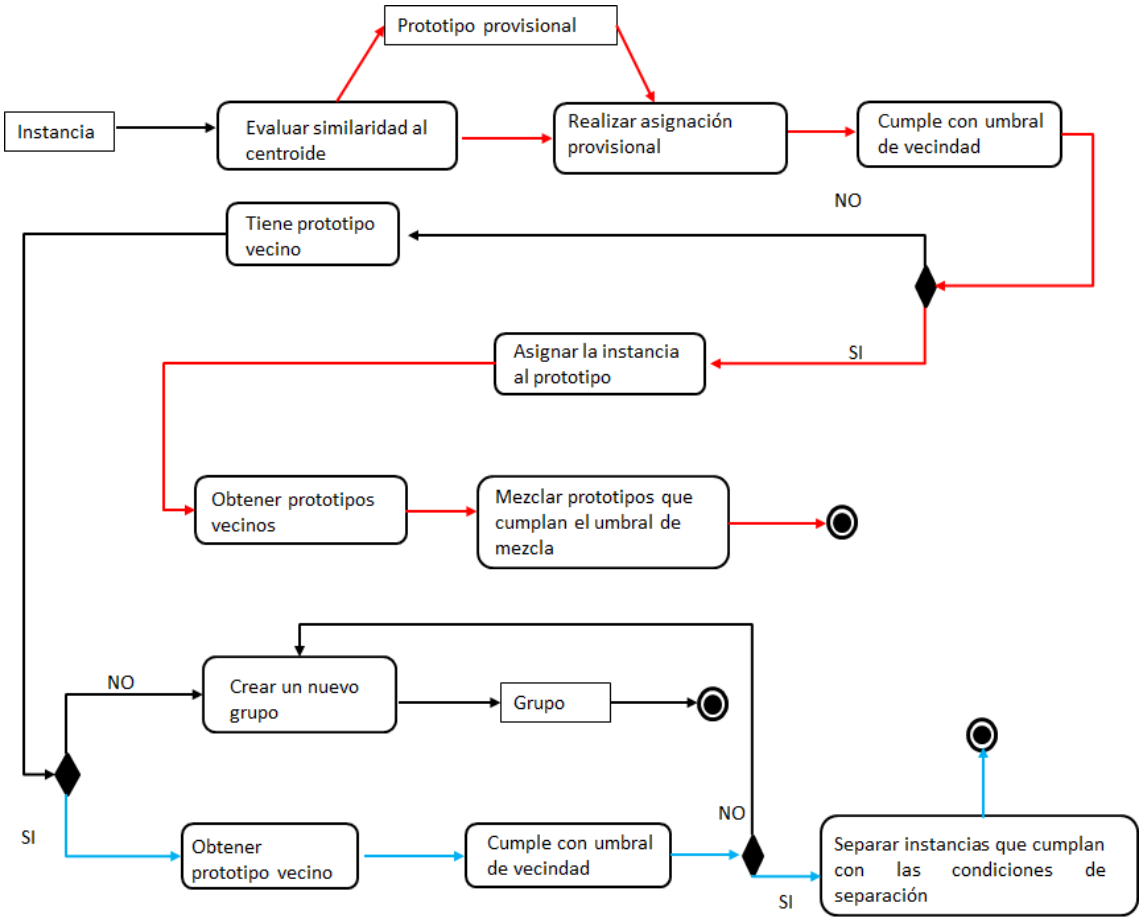


Figura 3.6. Diagrama de actividades de la etapa de clasificación.

Al culminar el proceso de creación de los grupos, se utiliza la métrica de densidad para verificar que los agrupamientos se realizaron correctamente. Es decir, se analizan las fronteras de los grupos con altas densidades para realizar mezcla de grupos donde corresponda. Este paso es opcional, es preferiblemente usado cuando se tienen bases de datos con altas densidades, y será detallado algorítmicamente más delante.

En caso de que un nuevo ejemplo en el sistema se desee agrupar después de haber culminado la etapa de entrenamiento, nuestra propuesta tiene la peculiaridad de que además de asignar el ejemplo al grupo más similar, está diseñada para que realice cambios sobre los prototipos; es decir, mezclas y separaciones, como en el caso de entrenamiento, siguiendo los mismos criterios.

c. Híbrido:

El caso híbrido que soporta la propuesta, es poder contener en una misma estructura la creación de clases y grupos, como se muestra en la Figura 3.7, que en la mayoría de los algoritmos estudiados no es permitido. En la Figura 3.7 se puede observar que existen ejemplos con etiquetas y ejemplos sin etiqueta, dichos ejemplos forman dos clases y un grupo bien definidos. También se puede observar que hay un ejemplo no etiquetado, que no pertenece a las clases y grupos creados. Se puede afirmar que el mismo no es similar a los prototipos existentes, dados los criterios previamente presentados, en consecuencia es necesaria la creación de un nuevo prototipo que lo represente. Este caso híbrido es planteado como resultado de la combinación de las actividades que se realizan para la etapa de clasificación y para la etapa de agrupamiento, para lo cual se propone una sincronización entre ambas etapas.

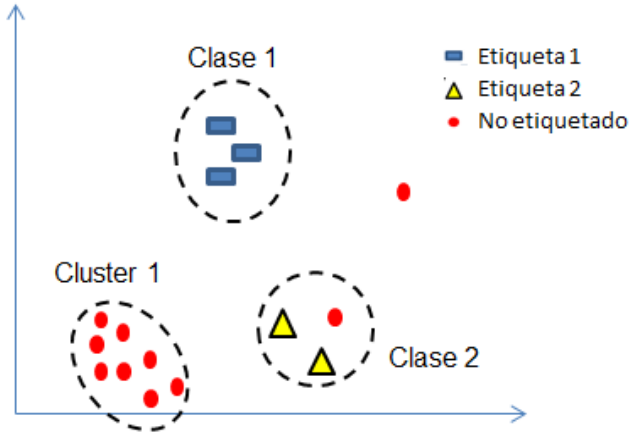


Figura 3.7. Caso híbrido que soportará la propuesta

El algoritmo va a ser capaz de diferenciar la creación de grupos o clases siguiendo el diagrama de actividades de la Figura 3.8. El mismo está compuesto tanto por las actividades que se realizan para el proceso de clasificación, como por las actividades que se realizan para el proceso de agrupamiento. Es capaz de diferenciar que vía tomar evaluando las características de la entrada y los prototipos.

En la Figura 3.8 se resaltan en color rojo las flechas que comparten ambos diagramas (clasificación y agrupamiento).Si la entrada es etiquetada se va a seguir el proceso de clasificación, si la entrada es no etiquetada se va por la vía del proceso de agrupamiento. Cabe destacar que en ambos diagramas (clasificación y agrupamiento) se determina el prototipo al cual va a ser asignado provisionalmente. Este prototipo provisional puede ser una clase o un grupo, por ello, el conocimiento del tipo de prototipo también definirá que vía se va a tomar. Es posible que en el ínterin de los procesos de clasificación/agrupamiento sea necesario moverse de un diagrama a otro, eso también dependerá del tipo de prototipo (clase o grupo) que se esté manipulando.

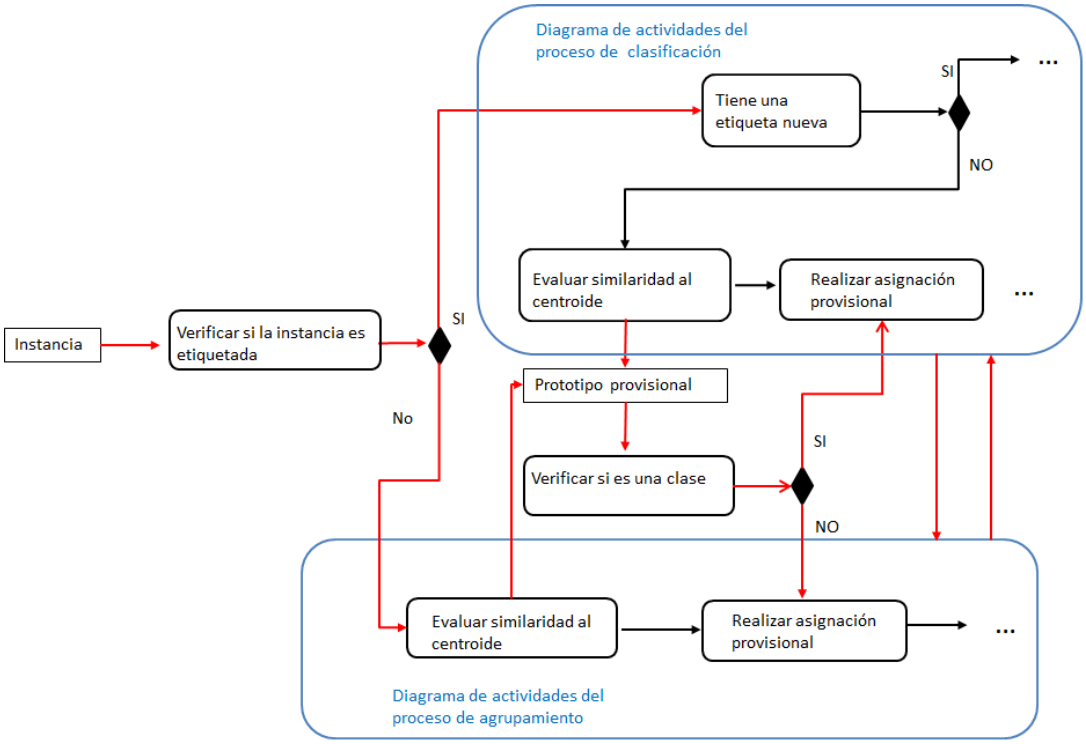


Figura 3.8. Diagrama de actividades del caso híbrido.

El diagrama de actividades expuesto, evidencia que si se está entrenando con una base de datos con todos sus ejemplos etiquetados, el proceso que se va a realizar es de clasificación pura. Si la base de datos tiene todos sus ejemplos no etiquetados, el proceso para entrenar va a ser de agrupamiento puro. En caso de tener una base de

datos con ejemplos etiquetados y no etiquetados, ambos planteamientos presentados de clasificación y agrupamiento trabajaran en armonía.

3.2.2.2. Macro algoritmo

El entrenamiento del modelo tiene como finalidad evaluar todas las instancias del conjunto de entrenamiento, y crear los prototipos que definirán el modelo del sistema. Dicha etapa está definida por el Macro Algoritmo 3.3.

Macro algoritmo 3.3. Entrenamiento del modelo

Entrada: Conjunto de entrenamiento *E*.

Cada instancia *e* se representa con una posición del arreglo *E*, esto es, *e= E(i)*

Procedimiento:

1. Repita desde *i = 0* hasta *tam(E)*
   - 1.1. Si es la primera vez que se introduce una entrada, se va a proceder de la siguiente manera según sea el caso:
     - 1.1.1. Entrada *e* etiquetada: Se crea una nueva clase
     - 1.1.2. Entrada *e* no etiquetada: Se crea un nuevo grupo
   - 1.2. Si no, si la entrada *e* es etiquetada con una etiqueta nueva en el sistema, crear una clase con dicha etiqueta
   - 1.3. Si no, si ya hay clases o grupos existentes:
     - 1.3.1. Calcular la similaridad del ejemplo *e* a los prototipos *P* (clases/ grupos) existentes
     - 1.3.2. Ubicar el prototipo con la mayor similaridad y asignar provisionalmente *e* a la clase/grupo correspondiente.
   - 1.4. Sino, crear un grupo con la entrada *e*
   - 1.5. Actualizar prototipos
2. Actualizar grupos
3. Fin

Salida: Prototipo(s) generados por el modelo

Para comprender el funcionamiento de la propuesta presentada en el macro algoritmo 3.3, se presenta a continuación con detalle sus ítems más resaltantes.

i. Actualizar prototipos (MA 3.3. ítem 1.5)

En este paso se evalúa la asignación provisional realizada en el ítem 1.4 del MA 3.3, en este sentido, se determinará si la entrada realmente pertenece al grupo/clase obtenida por medio del vecino más cercano, y si existen posibles mezclas entre clases/grupos o separaciones de instancias. Además, se estudia la etiqueta de la entrada y los cambios que ella puede causar a la clase/grupo en la que fue asignada.

Los principales pasos de este ítem están dados en el Macro Algoritmo 3.4, según sean las características de la entrada y la asignación realizada previamente por la similaridad con el centroide. Se ejecutarán 4 casos, que están compuestos por una serie de condiciones que permitirán cubrir las actividades propuestas en el modelo teórico. Cada caso se ejecuta cuando se presentan los siguientes escenarios:

- Caso 1: ocurre cuando se evalúa una entrada etiquetada conocida por el sistema, y dicha entrada es asignada provisionalmente a un grupo.
- Caso2: La entrada es etiquetada, conocida por el sistema, y se asignó provisionalmente sobre una clase con otra etiqueta (cruce de etiquetas).
- Caso 3: La entrada no es etiquetada, y fue clasificada provisionalmente en una clase.
- Caso 4: La entrada no es etiquetada y fue agrupada provisionalmente en un grupo.

La ocurrencia de cada caso dado en el Macro Algoritmo 3.4 desencadena la evaluación de una serie de condiciones, y en

consecuencia, la ejecución de acciones que afectan a los prototipos creados. Dichas condiciones y acciones serán especificadas más adelante.

Macro algoritmo 3.4. Actualizar prototipos

Entrada: Prototipos creados por el algoritmo *P* , instancia a evaluar *e* (entrada)

Procedimiento:

1. Para todos los casos, ubicar la clase/grupo *V* vecina(o) a la entrada *e*
2. Si *e* es una entrada etiquetada, con una etiqueta *j* conocida en el sistema, es clasificada en un grupo *i*.
   2.1. Ejecutar el Caso 1
3. Sino, si *e* es una entrada etiquetada conocida por el sistema, se asigna sobre una clase *i* con otra etiqueta (cruce de etiquetas).
   3.1. Ejecutar el Caso 2
4. Sino, si *e* es una entrada no etiquetada, clasificada en una clase
   4.1. Ejecutar el Caso 3
5. Sino, si *e* es una entrada no etiquetada, agrupada en un grupo
   3.1. Ejecutar el Caso 4

Salida: Prototipo(s) generado(s) o modificado(s) por el modelo

ii. Descripción de los casos de actualizar prototipos

Para la presentación detallada de los casos 1, 2,3 y 4, se va explicar la propuesta de cada uno de ellos algorítmicamente, y en algunos casos se presentan ejemplos gráficos que permitirán una mejor comprensión de los mismos. Para dichos ejemplos gráficos realizamos las siguientes suposiciones:

- Se representarán las fronteras de forma elíptica para separar las clases/grupos, aunque esta suposición no es necesariamente cierta ya que las clases/grupos pueden tener diferentes formas.

- Se considera que la notación para ejemplos no etiquetados es dada por el símbolo "●", mientras los etiquetados es cualquier otra representación simbólica.

- Los ejemplos son multidimensionales, pero para efectos prácticos se realiza la explicación de los mismos en un plano bidimensional que generaliza los demás casos.

Antes de presentar los algoritmos correspondientes a los casos 1, 2, 3 y 4, se presentan a continuación como se van a calcular algunas métricas y operaciones resaltantes que se van a utilizar:

1- Cálculo del centroide

Se calcula el centroide de cada clase/grupo utilizando la ec.(3.1) , calculando la media de todas las instancias pertenecientes a la clase/grupo correspondiente.

$$C_i = \frac{1}{n}\sum_{i=1}^{n} X_i = \frac{X_1 + X_2 + \cdots + X_n}{n} \qquad (3.1)$$

2- Función de similaridad

Para las primeras pruebas, la función para calcular la distancia entre una instancia $I = [i_1 \quad \dots \quad i_n]$ y otra instancia $A = [a_1 \quad \dots \quad a_n]$ es la distancia euclidiana, que viene dada por la ec.(3.2).

$$D(I,A) = \sqrt{(i_1 - a_1)^2 + (i_2 - a_2)^2 + \cdots + (i_n - a_n)^2} = \sqrt{\sum_{j=1}^{n} \left(i_j - a_j\right)^2} \qquad (3.2)$$

3- Densidad

La densidad de un grupo se calcula contabilizando la cantidad de elementos g con una etiqueta, dividido entre el total de elementos.

$$d = \frac{g}{N} \qquad (3.3)$$

4- Detectar la clase/grupo vecino de una instancia

Para detectar la clase/grupo vecino a una instancia j, se calcula la distancia euclideana entre j y los centroides de las clases/grupos existentes C_i , se ubica la distancia mínima, y la clase/grupo respectivo es el vecino, como se muestra en el Macro Algoritmo 3.5.

Macro algoritmo 3.5. Detectar una clase/grupo vecino

Entrada: Prototipos creados por el algoritmo *P* , instancia *j*

Cada prototipo se representa con una posición del arreglo *P*, esto es, *p= P(i)*

Procedimiento:

1. Repita desde *i = 0* hasta *tam(P)*
   - *1.1.* *C = obtener centroide de P(i)*
   - 1.2. *d = distancia(j,C)*
   - 1.3. Si *d < dmin* entonces
     - 1.3.1. *dmin =d*
     - 1.3.2. *V = P(i)*

Salida: Prototipo *V* vecino de *j*

5- Detectar la instancia vecina a otra instancia

En algunos casos va a ser necesario ubicar la instancia vecina en su propia clase/grupo, o a una instancia en otra clase/grupo. El procedimiento es igual que el caso anterior, la diferencia es que ya no se ubican los centroides de la clase, sino todas las instancias de una clase/grupo hasta conseguir la mínima distancia. En la Figura 3.6 se evidencia la instancia *i* de la Clase 1 vecina a la instancia *j,* y la instancia *e* vecina a *j* en su propio grupo. El Macro Algoritmo 3.6 evidencia los pasos para la detección de una instancia vecina a otra.

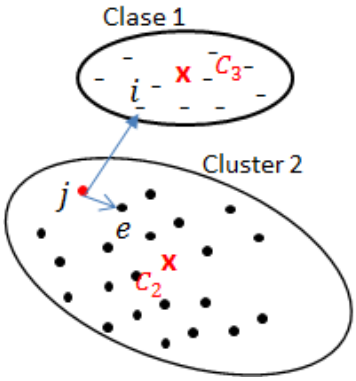


Figura 3.9. Ubicar la instancia vecina a otra instancia.

Macro algoritmo 3.6. Detectar una instancia vecina a otra en un prototipo

Entrada: Prototipo *p*, instancia *j*

Procedimiento:

1. Repita desde $i = 0$ hasta cantidad de instancias en *p*
    - 1.4. e = obtener instancia de *p* en la posición *i*
    - 1.5. $d = distancia(j,e)$
    - 1.6. Si $d < dmin$ entonces
        - 1.6.1. $dmin = d$
        - 1.6.2. $v = e$

Salida: Instancia *v* vecina de *j* en el prototipo *p*

6- Umbral de vecindad de instancia

Es un parámetro definido por el usuario, para iniciar el proceso o un valor por defecto. Cada grupo/clase tiene el mismo umbral de vecindad *uv* al inicio del proceso de entrenamiento, pero a medida que se van evaluando los ejemplos, cada clase/grupo modifica este parámetro. El umbral se modifica calculando la distancia promedio entre los ejemplos pertenecientes al grupo/clase utilizando la ec.(3.4), siendo x_i los ejemplos pertenecientes al grupo y *m* el total de ejemplos pertenecientes a un grupo/clase.

$$uv = \frac{\sum_i^m \sum_j^m |x_i - x_j|}{2m} \qquad (3.4)$$

7- Umbral de mezcla

Este umbral es menor que el umbral de vecindad y también podría ser modificado por el usuario; sin embargo, se coloca un valor fijo para cada clase/grupo proporcional al umbral de vecindad, de $0.6*uv$. Este valor se define de esa manera debido a que se desea restringir las mezclas de prototipos, de modo tal que solo ocurran cuando exista una distancia muy pequeña (más pequeña que el umbral de vecindad) entre los ejemplos de los prototipos que se deseen mezclar.

8- Separación simple de una instancia

La separación simple de una instancia *i* de una clase/grupo, se trata simplemente de sacarla del conjunto de elementos pertenecientes a la clase/grupo. Es decir, se elimina la instancia *i* de la clase/grupo a la cual pertenece.

9- Separación de una instancia no etiquetada en un grupo

Esta operación es utilizada solo por el agrupamiento, para separar una instancia *i* de un grupo (C_1), y la misma ser asignada en otro grupo (C_2), como se muestra en la Figura 3.10. Se sabe que cada grupo tiene su umbral de vecindad uv_j. Por lo tanto, dados C_1 y C_2, se toman las acciones dadas en el Macro Algoritmo 3.7.

Macro algoritmo 3.7. Separar una instancia de un grupo

Entrada: Prototipos C_1 y C_2

Instancia i

Procedimiento:

1. uv_1 = obtener umbral de C_1
2. uv_2 = obtener umbral de C_2
3. Si $uv_2 < uv_1$, entonces
   3.1. v = separar los vecinos del ejemplo i que tengan una distancia menor a uv_2 de C_2
   3.4. Asignar i y los vecinos v a C_2
4. Si no
   4.1. Realizar una separación simple del ejemplo i en C_1
   4.2. Asignar i a C_2

Salida: Prototipos C_1 y C_2

Esto quiere decir que si el umbral de vecindad del grupo 2 es menor del umbral del grupo 1, también se separan los ejemplos vecinos de i que sean menor al umbral del grupo 2. El resultado de esta operación está dado en la Figura 3.11.

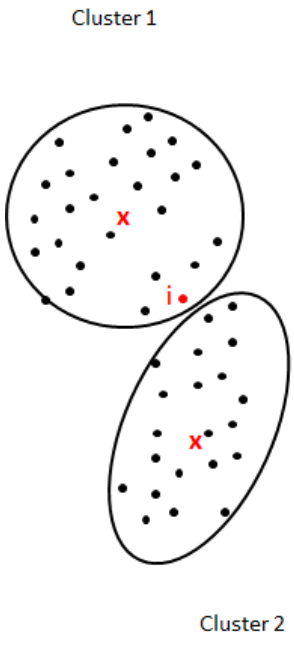
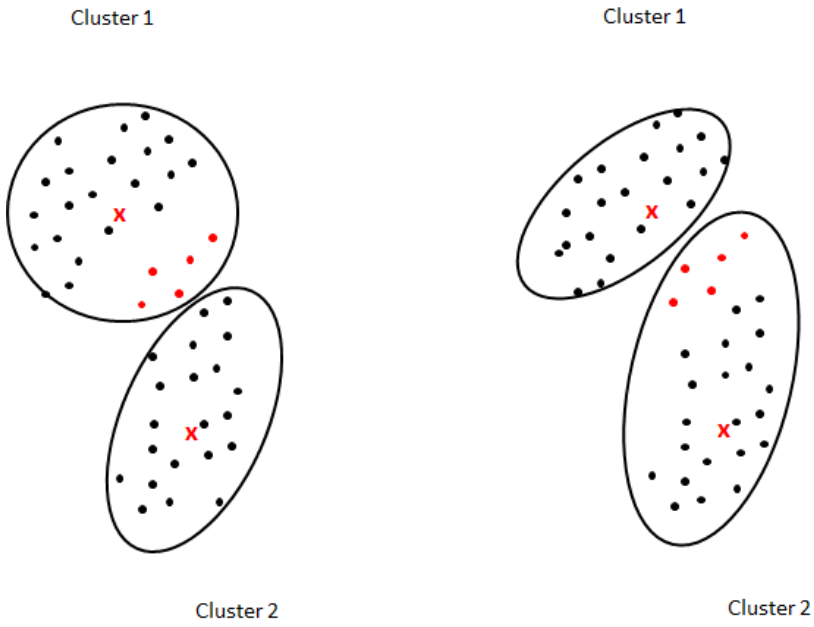


Figura 3.10. Separación del ejemplo i del grupo 1 al grupo 2.



(a) (b)

Figura 3.11. (a) Vecinos de *i* que cumplen con el umbral de vecindad del grupo 2 (b) Separación realizada.

Una vez hechas estas consideraciones, se presenta en detalle a continuación cada uno de los casos de la etapa de actualizar el sistema.

a. **Caso 1 (MA 3.4. ítem 2.1):** Entrada etiquetada, con una etiqueta *j* conocida en el sistema, agrupada en un grupo *i*. Este caso es evidenciado en la Figura 3.12, se puede observar que la etiqueta conocida está denotada con signo "-" y fue asignada en el grupo 1. Es importante resaltar que ya existe una clase con elementos del signo "-".

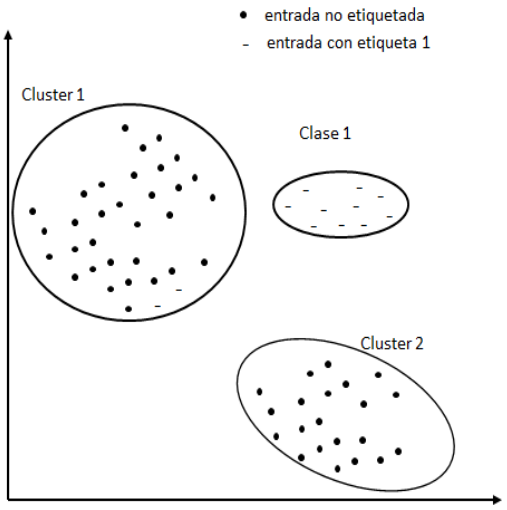


Figura 3.12. Ejemplo del caso 1; entrada etiquetada, con etiqueta "-" conocida por el sistema, agrupada en el grupo 1.

Se ubica la instancia vecino *v* de la entrada *j*, sobre todas las instancias del grupo *i*. Pueden ocurrir dos cosas: la primera es que la distancia sea menor al umbral de vecindad, en este caso se asigna la entrada y se verifica si hay posibles mezclas entre grupos. La segunda es que la distancia sea mayor, en este paso se trata de asignar la entrada a su prototipo vecino que tenga la misma etiqueta, como se muestra en la Figura 3.13, o se genera un grupo nuevo con esa instancia. El Macro Algoritmo 3.8 representa este caso.

Macro algoritmo 3.8. Caso 1

Entrada: Prototipo C_i

Instancia j

Procedimiento:

1. v = obtener instancia vecina de j en C_i
2. d = *distancia(v,j)*
3. uv_1 = obtener umbral de C_1
4. Si $d < uv_1$ *entonces*
   - 4.1. V = obtener todos los grupos vecinos de C_1
   - 4.2. C_1 = mezclar C_1 con los grupos vecinos V que cumplan con el umbral de mezcla
5. Si no
   - 5.1. Realizar separación simple de la instancia j de C_1
   - 5.2. V = obtener el prototipo vecino de j
   - 5.3. Si etiqueta(V) = etiqueta(j), entonces
     - 4.3.1. Asignar j a V
   - 5.4. Si no,
     - 5.4.1. Crear un nuevo grupo con la instancia j.

Salida: Prototipo(s) generado(s) o modificado(s) por este caso.

La creación de un nuevo grupo en el paso 4.4.1 del macro algoritmo 3.8, indica que a pesar de que la instancia es etiquetada y la etiqueta es conocida por el sistema, la misma no tiene suficientes características semejantes a su propia clase y esto podría significar que esta instancia representa una etiqueta errada, que puede generar ruido a las clases existentes. Es por ello, que nuestra propuesta es la creación de un nuevo grupo. Si a medida que se van evaluando más instancias y dicho grupo creado se va asemejando a la clase original a quien pertenecían los ejemplos etiquetados, el algoritmo notará este comportamiento y realizará la mezcla entre la clase y el grupo mediante el paso 3.2.

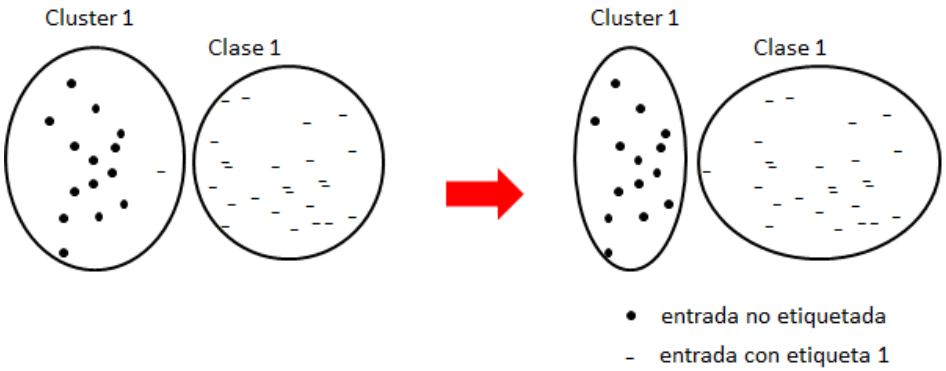


Figura 3.13. Separación de la instancia *j* y unión con su clase correspondiente.

b. **Caso 2 (MA 3.4. ítem 3.1):** Entrada etiquetada *j* conocida por el sistema, se asigna sobre una clase *i* con otra etiqueta (cruce de etiquetas). En la Figura 3.14 se muestran las características de este caso.

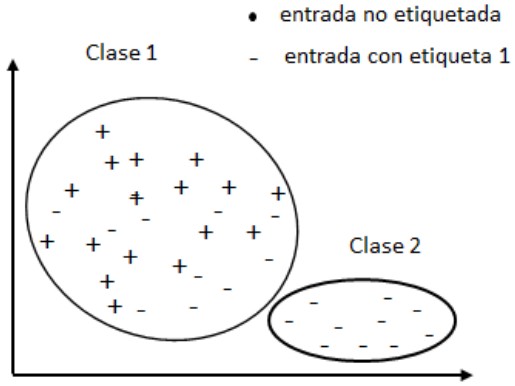


Figura 3.14. Ejemplo del caso 2; entrada etiquetada "-", con etiqueta conocida por el sistema, clasificada en una clase con otra etiqueta "+".

Este escenario puede representar un error de clasificación generado por la métrica de similaridad con el centroide, es por ello que se evalúa igualmente la relación de la instancia con su vecino más cercano. Si la distancia entre la instancia y el vecino cumple con el umbral de vecindad, la asignación fue correcta (véase el Macro Algoritmo 3.9). Por otro lado, la asignación sucesiva de ejemplos con otra etiqueta a una clase sugiere un problema con los datos, comúnmente denominados como datos desequilibrados (He H. and

Garcia E., 2009), o un solapamiento entre clases que no está permitido en los algoritmos de clasificación. Es por ello, que se plantea con esta propuesta la posibilidad de una mezcla de clases cuando las mismas están solapadas. La etiqueta después de realizar la mezcla es la concatenación de las etiquetas de las clases que fueron mezcladas.

Macro algoritmo 3.9. Caso 2

Entrada: Prototipo C_i

Instancia j

Procedimiento:

1. v = obtener instancia vecina de j en C_i
2. d = *distancia(v,j)*
3. uv_l = obtener umbral de C_l
4. Si $d < uv_l$, entonces
   - 4.1. *den* = obtener densidad de la etiqueta de j en C_l
   - 4.2. Si *den* es mayor que el umbral de densidad y más de la mitad de los ejemplos de la base de datos han sido evaluados
     - 4.2.1. V = obtener el prototipo vecino de j
     - 4.2.2. v = obtener instancia vecina de j en V
     - 4.2.3. d = *distancia(v,j)*
     - 4.2.4. Si etiqueta(V) = etiqueta*(j)* y $d <$ umbral de mezcla.
       - 4.2.4.1. C_l= mezclar(V, C_l)
       - 4.2.4.2. etiqueta(C_l)= etiqueta(C_l)+ etiqueta(V)
5. Si no
   - 5.1. Separar instancia j de C_l
   - 5.2. V = obtener el prototipo vecino de j
   - 5.3. Si etiqueta(V) = etiqueta(j), entonces
     - 5.3.1. Asignar j a V
   - 5.4. Si no,
     - 5.4.1. Crear un nuevo grupo con la instancia j.

Salida: Prototipo(s) generado(s) o modificado(s) por este caso.

En la Figura 3.15 se observa el resultado de realizar la acción del paso 3.2.2.1 del macro algoritmo 3.9, y en la Figura 3.16 se muestra el resultado de separar la instancia después de haber sido evaluada la condición 4.3. El caso dado en la Figura 3.15 no es común en los problemas de clasificación, normalmente se pueden diferenciar claramente las clases en un problema.

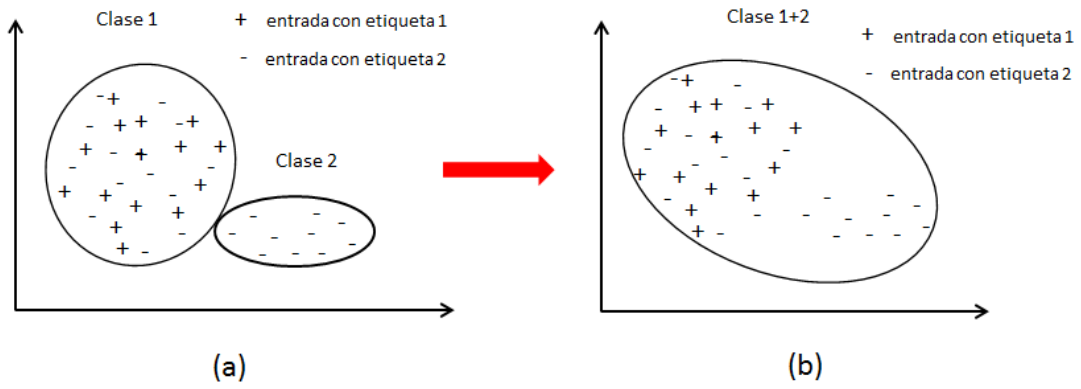


Figura 3.15. (a) Ejemplo del caso 2; cuando la densidad de los elementos etiquetados "-" es grande, y además, a la clase 2 al cual pertenece la entrada es vecina. (b) Ejemplo del caso 2, acción tomada.

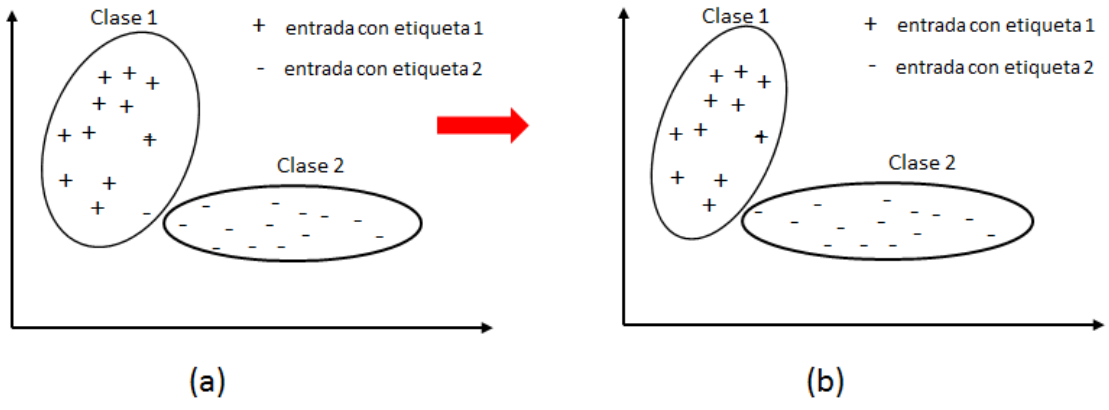


Figura 3.16. (a) Ejemplo del caso 2; cuando la densidad de los elementos etiquetados "-" es pequeña, y además, a la clase 2 al cual pertenece la entrada es vecina. (b) Ejemplo del caso 2, acción tomada.

c. **Caso 3 (MA 3.4. ítem 4.1):** Entrada no etiquetada clasificada en una clase, este caso es evidenciado en la Figura 3.17.

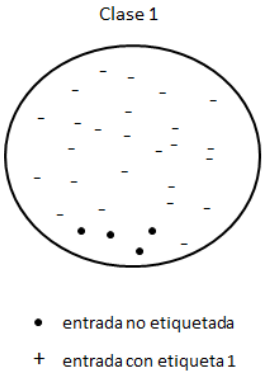


Figura 3.17. Ejemplo del caso 3, entrada no etiquetada clasificada en una clase.

Al igual que los casos anteriores, se evalúa la relación de la instancia con su vecino más cercano en la clase a la cual fue asignada. Si cumple con el umbral de vecindad la instancia se mantiene en dicha clase y se etiqueta; además; se buscan los grupos vecinos con la finalidad de detectar posibles mezclas entre clase-grupo; en caso que hayan mezclas, la etiqueta de la clase se mantiene. El Macro Algoritmo 3.10 representa este caso.

Macro algoritmo 3.10. Caso 3

Entrada: Prototipo C_i

Instancia j

Procedimiento:

1. v = obtener instancia vecina de j en C_i
2. d = *distancia(v,j)*
3. uv_1 = obtener el umbral C_1
4. Si $d < uv_1$ *entonces*
   - 4.1. V = obtener todos los grupos vecinos de C_1
   - 4.2. C_1 = mezclar C_1 con los grupos vecinos V que cumplan con el umbral de mezcla
5. Si no
   - 5.1. Separar instancia j de C_1
   - 5.2. V = obtener el prototipo vecino j
   - 5.3. v = obtener instancia vecina de j en V
   - 5.5. d = *distancia(v,j)*
   - 5.6. uv_2 = obtener el umbral V
   - 5.7. Si $d < uv_2$, entonces
     - 4.5.1. Asignar j a V
   - 5.8. Si no,
     - 5.8.1. Crear un nuevo grupo con la instancia j.

Salida: Prototipo(s) generado(s) o modificado(s) por este caso.

d. **Caso 4(MA 3.4. ítem 5.1):** Entrada no etiquetada, agrupada en un grupo, este caso es evidenciado en la Figura 3.18.

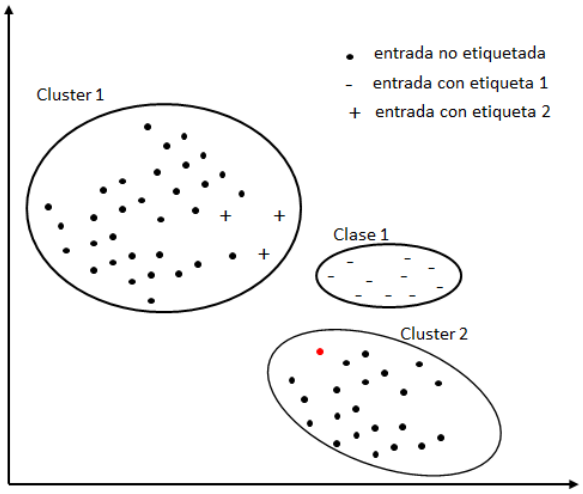


Figura 3.18. Ejemplo del caso 4; entrada no etiquetada, agrupada en un grupo.

Este escenario representa el proceso de agrupamiento no supervisado como tal, el mismo es tratado como los casos anteriores. Se busca el vecino más cercano y se valida la asignación, se detectan las posibles mezclas grupo-grupo o grupo-clase (si existen clases creadas en el sistema). En caso de que la instancia no cumpla con el umbral de vecindad se separa del grupo al cual fue asignado por similaridad con el centroide, esta separación puede traer como consecuencia la separación de más instancias debido al planteamiento dado en el MA 3.7.

Macro algoritmo 3.11. Caso 4

Entrada: Prototipo C_i

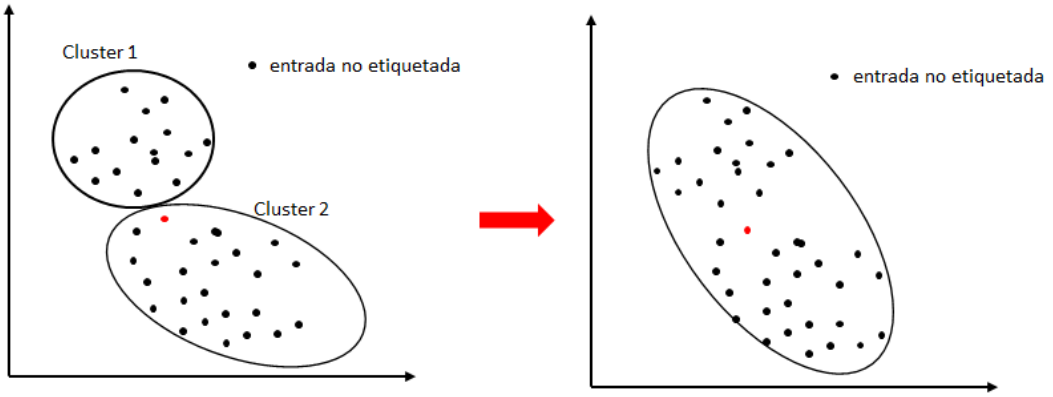
Instancia j

Procedimiento:

1. v = obtener instancia vecina de j en C_i
2. d = *distancia(v,j)*
3. uv_1 = *obtener el umbral de* C_1
4. Si $d < uv_1$ *entonces*
   - 4.1. V = obtener todos los grupos vecinos de C_1
   - 4.2. C_1 = mezclar C_1 con los grupos vecinos V que cumplan con el umbral de mezcla
5. Si no
   - 5.1. V = obtener el prototipo vecino de j
   - 5.2. v = obtener instancia vecina de j en V
   - 5.3. d = *distancia(v,j)*
   - 5.4. uv_2 = *obtener el umbral de* V
   - 5.5. Si $d < uv_2$ *entonces*
     - *5.5.1.* Separar instancia j y vecinos de C_1
     - 5.5.2. Asignar j a V
   - 5.6. Si no,
     - 5.6.1.Separar instancia j de C_1
     - 5.6.2. Crear un nuevo grupo con la instancia j.

Salida: Prototipo(s) generado(s) o modificado(s) por este caso.

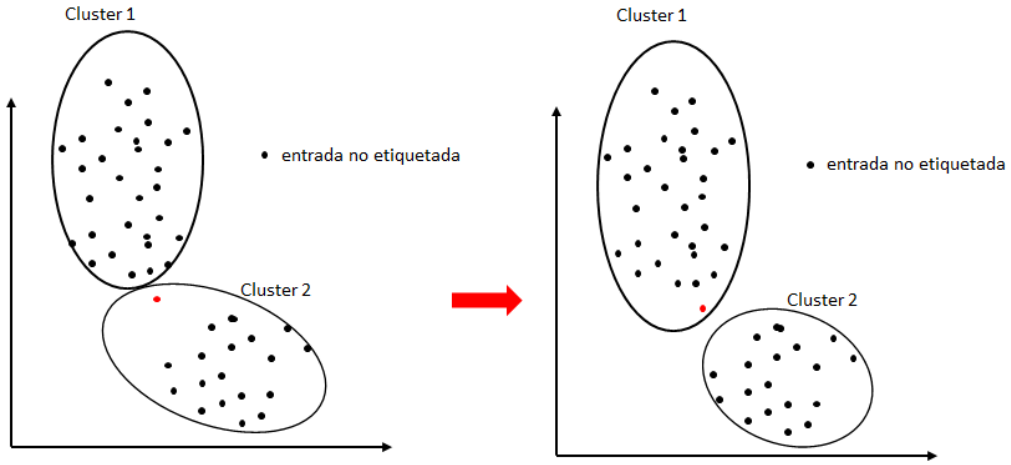
En caso que se haya validado que la instancia pertenece al grupo donde fue asignado, como se muestra en la Figura 3.19-(a), y además, que exista un grupo vecino que cumpla con el umbral de mezcla, el resultado que se genera se muestra en la Figura 3.19-(b).



(a) (b)

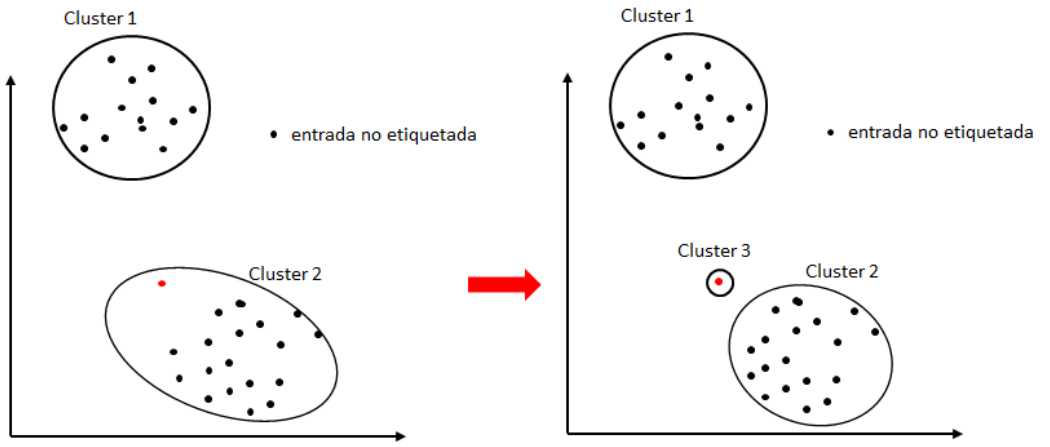
Figura 3.19. (a) Ejemplo del caso 4, se cumple la condición de mezcla (b) Resultado que se obtiene después de haber mezclado los grupos.

Si la asignación por medio del vecino más cercano no fue validada, el ejemplo se separa del grupo y se asigna a su grupo vecino si cumple con el umbral de vecindad, como se muestra en las Figuras 3.20-(a) y (b). En caso de que no cumpla con el umbral de vecindad del grupo vecino, separa la instancia y se crea un nuevo grupo como se muestra en las Figuras 3.21-(a) y (b)



(a) (b)

Figura 3.20. (a) Ejemplo del caso 4, separación de la entrada resaltada con rojo del grupo 2 (b) Resultado de separar la instancia resaltada con rojo del grupo 2 y de asignarla al grupo 1.



(a) (b)

Figura 3.21. (a) Ejemplo del caso 4, separación de la entrada resaltada con rojo del grupo 2 (b) Creación de un nuevo grupo con la entrada resaltada con rojo.

iii. Actualizar grupos(MA 3.3. ítem 2)

Este paso es opcional y es óptimo utilizarlo cuando se cuentan con grandes bases de datos con densidades altas, debido a que permite mejorar los resultados obtenidos por la fase de agrupamiento en el entrenamiento. Esta etapa tiene como finalidad detectar los grupos que deberían estar unidos por la alta densidad de ejemplos en sus fronteras. Es por ello que se recorre cada prototipo, como se muestra en el Macro Algoritmo 3.12, y se decide si se realiza una mezcla o no con cualquier otro prototipo creado por los criterios dados en el Macro Algoritmo 3.13.

Macro algoritmo 3.12. Actualizar grupos

Entrada: Prototipos creados por el algoritmo *P*

Procedimiento:

1. Repita desde $i = 0$ hasta *tam(P)*
   1.1. Si *P(i)* no ha sido visitado y es un grupo
      1.1.1. *P(i)* = mezclar(*P(i)*)

Salida: Prototipo(s) generado(s) o modificado(s) por el modelo

Macro algoritmo 3.13. Mezcla de grupos

Entrada: Prototipo a evaluar *p*

Procedimiento: mezclar(*p*)

1. Repita mientras todos los grupos vecinos de *p* sean visitados
    1.1. *v* = buscar grupo vecino de *p*
    1.2. Si existe un área densa en un radio r con densidad mayor a un umbral, entre *v* y *p*
        1.2.1. Mezclar los grupo *v* y *p*
        1.2.2. Retornar el grupo nuevo

Salida: Retorna el prototipo *p* como resultado de hacer la(s) mezcla(s) o el prototipo *p* cuando no hubo mezcla

Para ilustrar los efectos que ocasiona esta etapa opcional, supóngase que se obtienen los grupos de la Figura 3.22 una vez culminada la fase de entrenamiento. En la Figura 3.22, los puntos rojos son grupos con gran cantidad de ejemplos concentrados en una zona y con distancia muy pequeñas entre sí.

Para decidir qué grupos deben mezclarse se estudia la densidad en una radio *r,* como se muestra en la Figura 3.23, los círculos punteados en color rojo representan radios que se están estudiando. En dicho radio, los grupos que estén rodeados de altas densidades con otros grupos se deben mezclar. Por ejemplo, el grupo 7 está rodeado en sus fronteras por alta densidad de ejemplos no etiquetados de los ejemplos del grupo 6, es por ello que se decide realizar la mezcla de los mismos. Caso similar ocurre con los grupos 1, grupo 2, grupo 4 y grupo 5, que cumple la condición de mezcla. El resultado de aplicar esta operación viene dado por la Figura 3.24.

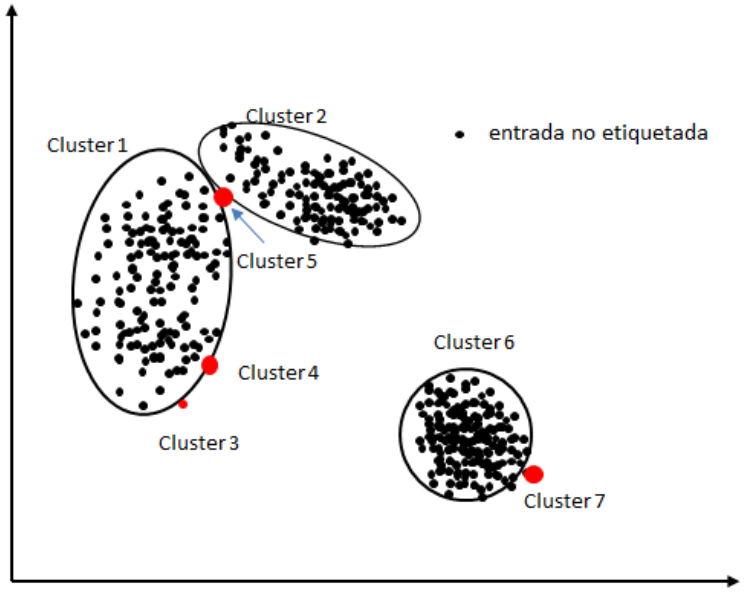


Figura 3.22. Resultado obtenido con el modelo híbrido una vez culminada la fase de entrenamiento.

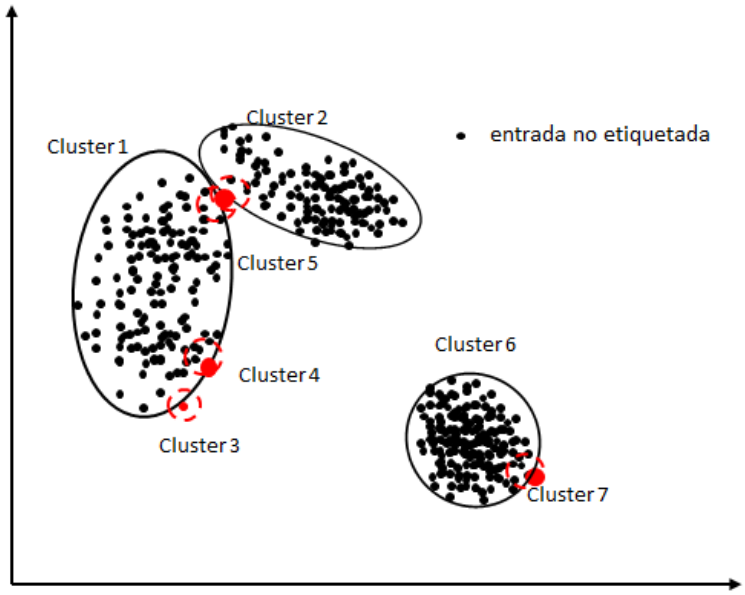


Figura 3.23. Detección de los grupos candidatos para ser mezclados.

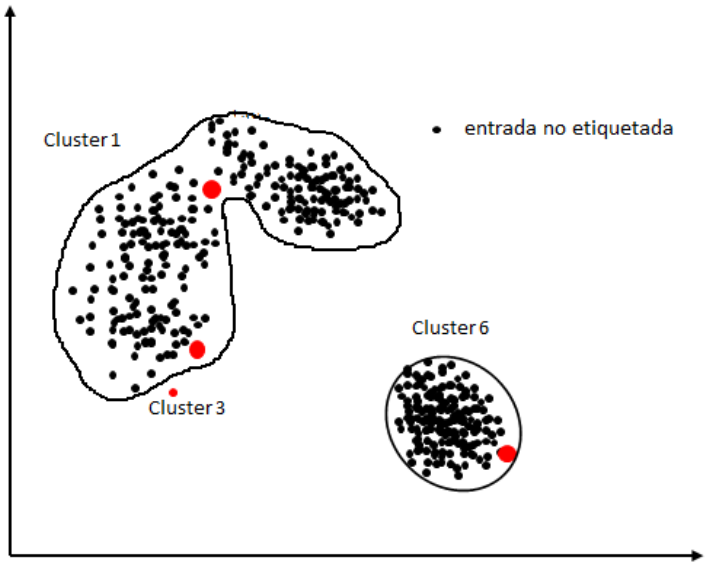


Figura 3.24. Resultado final al aplicar las acciones definidas para actualizar grupos.

3.2.3. Evaluar los prototipos (MA 3.1. ítem 3)

Para evaluar el modelo generado por la propuesta, se pueden utilizar las métricas descritas en los puntos 2.2.2 para clasificación y 2.3.1 para agrupamiento. A continuación se muestran las métricas seleccionadas, para cada escenario que puede resolver nuestra propuesta:

a. Clasificación: Se plantea la métrica de precisión como medida principal para la evaluación de las clases creadas, ya que permite medir el desempeño del proceso de clasificación, y así concluir si la construcción de las clases fue la correcta (cuando la precisión es cercana a 1) o incorrecta (cuando la precisión es cercana a 0).

b. Agrupamiento: Se utilizan dos índices internos para validar los agrupamientos, estos son:

    i. Cohesión (Co): para determinar la cohesión de cada grupo, se determina la distancia entre cada ejemplo del grupo y su centroide, como se muestra en la ec. 3.5, siendo c_i el

centroide del grupo i, n la cantidad de ejemplos que tiene el grupo, y X_{ij} los ejemplos pertenecientes al grupo i

$$Co = \frac{1}{n}\sum_{j=1}^{n}|c_i - X_{ij}| \qquad (3.5)$$

ii. Separación: el cálculo de la separación entre dos grupos viene dado por la diferencia entre sus centroides.

$$Sep = |c_i - c_j| \qquad (3.6)$$

Una cohesión grande podría significar que los datos están muy dispersos o alejados del centroide, por otro lado, una cohesión pequeña podría indicar que los datos están bien compactados en el prototipo. Una separación grande entre los prototipos podría indicar una formación adecuada de los agrupamientos, una separación pequeña podría indicar solapamiento entre los prototipos. Vemos así que en general, la cohesión y separación nos podrían indicar ciertos resultados, pero no hay manera de asegurar que los grupos fueron correctamente creados ya que son valores relativos (Tan P., et al., 2005).

Debido a que la cohesión y la separación son métricas que dan como resultado una distancia, y dichas distancias podrían proporcionar o no información de los agrupamientos formados, se plantea adicionalmente la métrica de la ec. (3.7), que combina el promedio de ambas (Co y Sep), con la finalidad de entender el resultado generado por estas métricas. Co_μ es el promedio de las cohesiones de todos los grupos y Sep_μ es el promedio de las separaciones entre todos los grupos. Esta evaluación del agrupamiento es una propuesta del trabajo inspirada en los índices internos.

$$M = 1 - \frac{Co_\mu}{Sep_\mu} \qquad (3.7)$$

Si $M<0$ la cohesión es mayor que la separación, las agrupaciones podrían estar formadas incorrectamente.

Si $M \approx 1$ la cohesión es más pequeña que la separación, las agrupaciones están bien formadas.

Si $M \approx 0$ no es concluyente, las agrupaciones podrían estar bien formadas o no.

c. Híbrido: Para evaluar numéricamente la eficiencia del algoritmo de los casos híbridos, se propone en el presente trabajo una métrica que combina las métricas de clasificación y agrupamiento. Se plantea calcular la precisión para las clases creadas, la cohesión y separación para los grupos, y finalmente se calcula una suma ponderada entre ambas dada por la ec. (3.8). Esta métrica tiene como finalidad obtener cuantitativamente un número con el que se podrá concluir si los prototipos (clases o grupos) fueron creados correctamente.

$$H = \rho.p + \beta.M \qquad (3.8)$$

ρ es la fracción de las clases creadas en el sistema, y viene dado por la ec. (3.9), N_C es la cantidad de clases creadas por el algoritmo y N_{Cl} es la cantidad de grupos creados por el algoritmo.

$$\rho = \frac{N_C}{N_{Cl} + N_C} \qquad (3.9)$$

Así mismo, β es la fracción de grupos creados en el sistema, y viene dado por la ec. (3.10).

$$\beta = \frac{N_{Cl}}{N_C + N_{Cl}} \qquad (3.10)$$

Sabemos que la precisión p toma valores entre 0 y 1, siendo 1 la precisión más elevada que puede proporcionar un algoritmo. M puede tomar valores menores que 0 y entre 0 y1. Por lo tanto, el rango de valores que puede tomar H está definido por:

$H<0$ ocurre cuando $\rho.p < \beta.M$ y $M<0$, los prototipos creados podrían estar formadas incorrectamente.

H cercanos a 1, los prototipos creados están bien formados.

H cercanos a 0, en este caso si $\rho \gg \beta$ quiere decir que se crearon más clases que grupos y la precisión es pequeña, por lo tanto los prototipos creados están formados incorrectamente. Si $\rho \ll \beta$ se crearon más grupos que clases y la métrica no es concluyente, requiere de una revisión de las características de los datos para observar si existen anomalías por las cuales el modelo no es capaz de crear grupos adecuadamente.

Se plantean dos tipos de evaluación de los prototipos, la primera consiste en evaluar el entrenamiento del modelo, y la segunda evaluar la etapa de funcionamiento cuando se cuentan con datos de prueba.

3.2.3.1. Evaluación de la etapa de entrenamiento

El Macro Algoritmo 3.14 describe los pasos a realizar para la evaluación del entrenamiento, que se traducen al cálculo de las métricas previamente definidas y a mostrar los valores obtenidos.

Macro Algoritmo 3.14. Evaluación del entrenamiento

- **Entradas:** Prototipos *P*, conjunto de entrenamiento *E*
- **Procedimiento:**
  1. Calcular *p* de las clases con *P* y *E*
  2. Calcular M de los grupos con *P*
  3. Calcular *H*
  4. Generar resumen de la evaluación
  5. Fin
- **Salida:** Resumen de evaluación

3.2.3.1. Evaluación de funcionamiento del sistema

La diferencia con la etapa de entrenamiento es que se cuenta con datos de prueba que tienen etiquetas, por lo tanto, se puede realizar una correspondencia entre los resultados obtenidos por el algoritmo propuesto y por los valores esperados reales del conjunto de prueba. La evaluación del funcionamiento en línea es válida solo para la clasificación, ya que en agrupamiento no se cuenta con las etiquetas de los ejemplos. El Macro Algoritmo 3.14 describe los pasos a realizar para la evaluación.

Macro Algoritmo 3.15. Evaluación del funcionamiento

- **Entradas:** Etiquetas reales de los ejemplos de prueba *L*, Etiquetas generadas por el algoritmo después de evaluar los ejemplos de prueba *Rl*
- **Procedimiento:**
  1. Calcular *p* de las clases con *L* y *Rl*
  2. Generar resumen de la evaluación
  3. Fin
- **Salida:** Resumen de evolución

Debido a que no es posible evaluar el agrupamiento en línea, basta con observar los resultados obtenidos en la evaluación que se realiza para la etapa

de entrenamiento. De allí se puede concluir qué tan bien el algoritmo crea los grupos. Cabe destacar que una vez entrenado el sistema, el funcionamiento en línea se comporta como un algoritmo no supervisado.

3.2.4. Etapa de funcionamiento del sistema (MA 3.1. ítem 4)

El objetivo del funcionamiento del algoritmo es clasificar o agrupar, según sea el caso, una entrada no etiquetada. El resultado de este proceso viene dado por la pertenencia de un nuevo ejemplo a un prototipo, dicho prototipo puede ser tanto una clase/grupo ya existente en el sistema, como un nuevo prototipo creado por esta entrada. Esta etapa viene dada por el Macro Algoritmo 3.16.

La etapa de actualización (ítem 4) de prototipos en el funcionamiento del sistema cambia con respecto a la fase de entrenamiento, ya que solo hay que considerar que las entradas siempre serán no etiquetadas. Las entradas no son etiquetadas debido a que son nuevos ejemplos desconocidos por el sistema, de los cuales no se conocen las clases, o en su defecto a los grupos, a los que pertenecen. El sistema, en el funcionamiento, debe ser capaz de detectar las similitudes entre el ejemplo de entrada y los prototipos ya existentes, utilizando los planteamientos de los casos 3 y 4 dados en la fase de entrenamiento para actualizar los prototipos (MA 3.10 y MA 3.11). En caso de no existir similitudes entre la entrada y los prototipos existentes, el algoritmo está diseñado para la creación e integración de nuevos prototipos al modelo.

Por otro lado, en caso de que ya se haya entrenado el modelo y se quieran integrar nuevos ejemplos etiquetados al mismo, se tienen las siguientes opciones:

- La primera es el reentrenamiento del modelo, integrando las entradas etiquetadas a la base de datos de entrenamiento.
- La segunda se basa en remover las etiquetas de las entradas y evaluarlas en el modelo en la etapa de funcionamiento. El sistema debe ser capaz de crear un nuevo grupo con estas entradas, y luego de ser creado, el algoritmo propuesto permite etiquetar grupos existentes y convertirlos en una clase.

Macro algoritmo 3.16. Funcionamiento

Entrada: Entrada a evaluar *e*

Procedimiento:

1. Leer el ejemplo *e* introducido
2. Calcular similaridad de *e* a las clases/grupos existentes.
3. Ubicar las clase/cluster con la mayor similaridad y asignar la entrada *e* a la clase/grupo correspondiente.
4. Actualizar prototipos
5. Generar Resultado

Salida: Prototipo *p* al cual fue asignado a la entrada por el algoritmo

Es importante señalar, que en el proceso de funcionamiento del algoritmo tiene la capacidad de crear nuevos prototipos, como consecuencia de las operaciones que se realizan en los casos 3 y 4. Esta capacidad es muy poco común en los algoritmos de MD.

Capítulo 4

Implementación, experimentos y análisis de resultados del algoritmo híbrido

Una vez definidas las bases teóricas que constituyen el algoritmo híbrido, se procede a la implementación del mismo, seguidamente se realizan las pruebas necesarias que permiten validar nuestra propuesta.

4.1. Implementación del algoritmo híbrido

Para realizar una abstracción del planteamiento realizado en el Capítulo 3, se utiliza la programación orientada a objetos y los diagramas de clases UML para mostrar las relaciones entre las clases (detalles en Apéndice A).

4.1.8. Diagrama UML del sistema

a. Entrada al sistema

   La entrada a un problema de MD viene dado por instancias, cada instancia se representa con una clase (llamada *Instancia*) que está compuesta por sus atributos y por su etiqueta. A su vez, el conjunto completo de instancias será representado por la clase *Instancias*, como se muestra en la Figura 4.1.

La clase *Data* se encarga de cargar el archivo y de realizar la lectura adecuada de los datos; además, se encarga de transformar el formato de los atributos y las etiquetas en objetos *Instancia*. También se pueden realizar análisis estadísticos sobre la base de datos como obtener la varianza, el promedio de la distancia entre los atributos, entre otros (estas son funciones opcionales del sistema). El diagrama de la Figura 4.1 muestra la relación entre las clases *Instancia*, *Instancias* y *Data*.

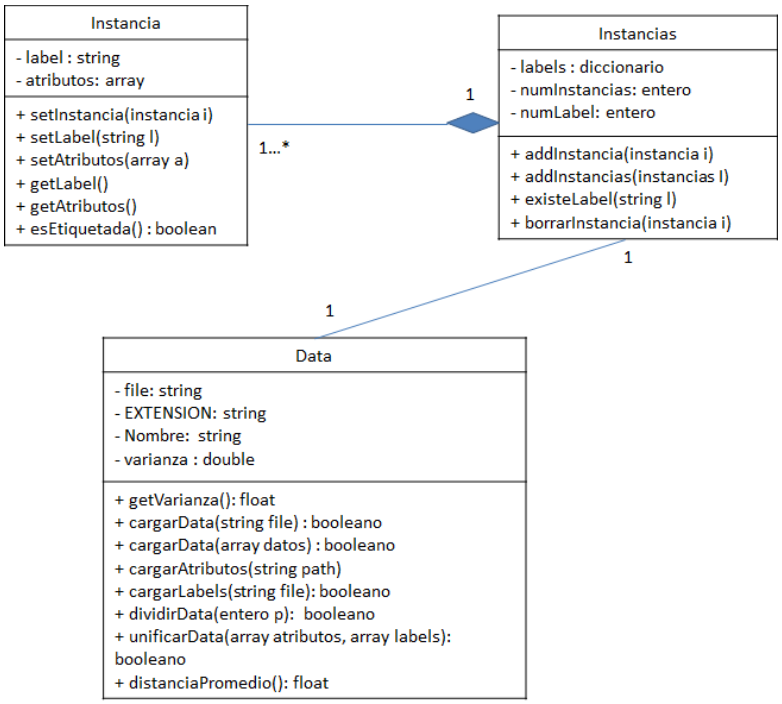


Figura 4.1. Diagrama de clases de entradas del sistema.

b. Proceso:

Se define la clase *Prototipo,* que contiene toda la información necesaria para caracterizar un grupo/clase, como se muestra en la Figura 4.2. A dicha clase se le adiciona como atributo un árbol de distancias entre las instancias para una búsqueda más eficiente del vecino más cercano. En esta clase se implementan los algoritmos de creación, eliminación, mezcla de prototipos, además de, separación de instancias en un prototipo y búsqueda del vecino

más cercano. Entre los macro algoritmos implementados en esta clase se encuentran el 3.6 y 3.7 descritos en el Capítulo 3.

La clase *Algoritmo* es la encargada de la construcción, modificación y manipulación de los prototipos, la misma utiliza las instancias provenientes de la clase *Data* para la creación de los prototipos. Entre los macro algoritmos implementados en esta clase se encuentran el 3.3, 3.4, 3.8, 3.9, 3.10, 3.11, 3.12 y 3.13 descritos en el Capítulo 3. El diagrama relacionado a estas clases es dado en la Figura 4.2.

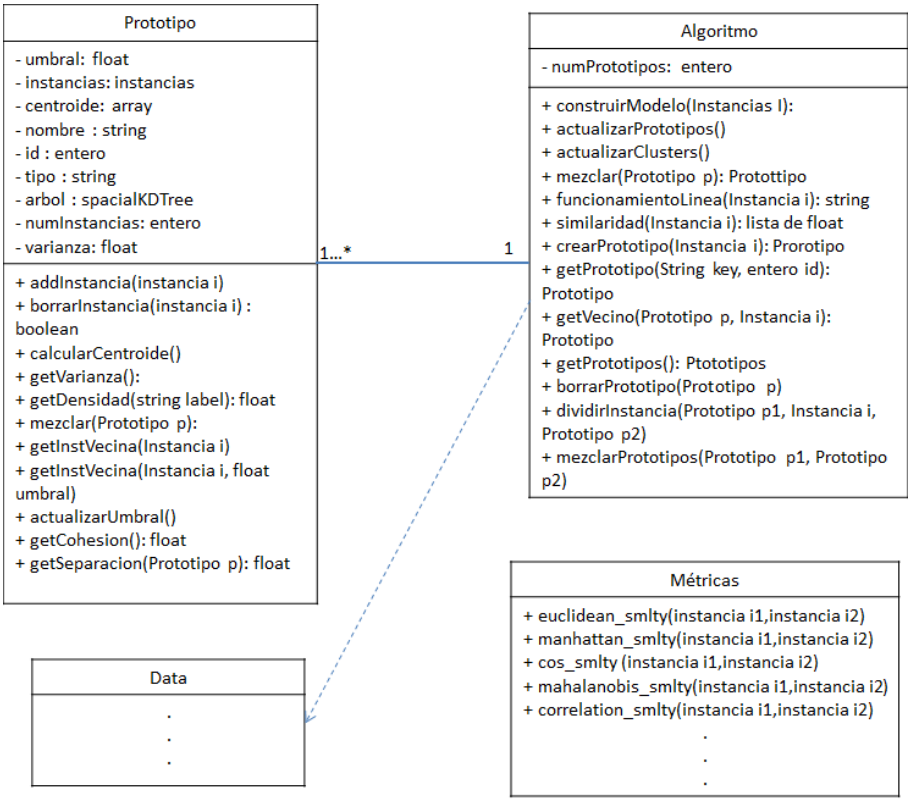


Figura 4.2. Diagrama de clases del proceso del sistema.

c. Salida del sistema

La salida del algoritmo puede verse como los prototipos creados por el algoritmo, pero también la salida se puede representar gráficamente y con la evaluación de los prototipos. La clase *Evaluación* implementa los macro algoritmos 3.14 y 3.15, además del cálculo de las diferentes métricas utilizadas por dichos macro algoritmos. La clase *PCAadaptado* permite graficar

los prototipos en un plano bidimensional, reduciendo las instancias de n-dimensiones a 2D utilizando las bases teóricas dadas en la sección 2.5. El diagrama de clases relacionado a esta etapa viene dado por la Figura 4.3.

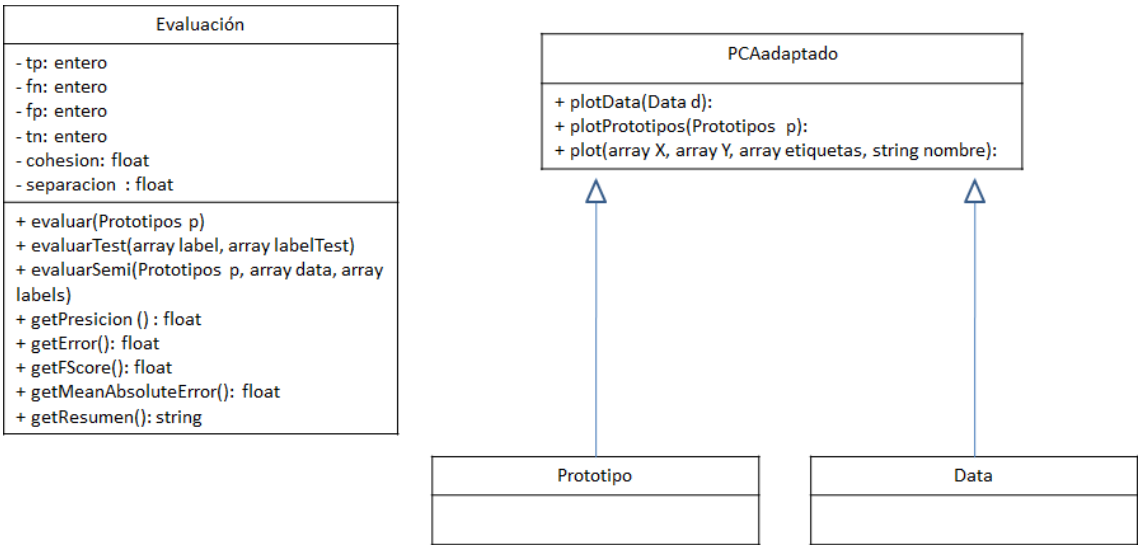


Figura 4.3. Diagrama de clases de la salida del sistema.

El diagrama completo del sistema viene dado por la Figura 4.4. El proceso inicia con la creación de las instancias por medio de la clase *Data*, la clase *Algoritmo* utiliza la información generada de la base de datos para la creación de los prototipos, y seguidamente se muestran los resultados obtenidos con el uso de las clases *PCAadaptado* y *Evaluación*.

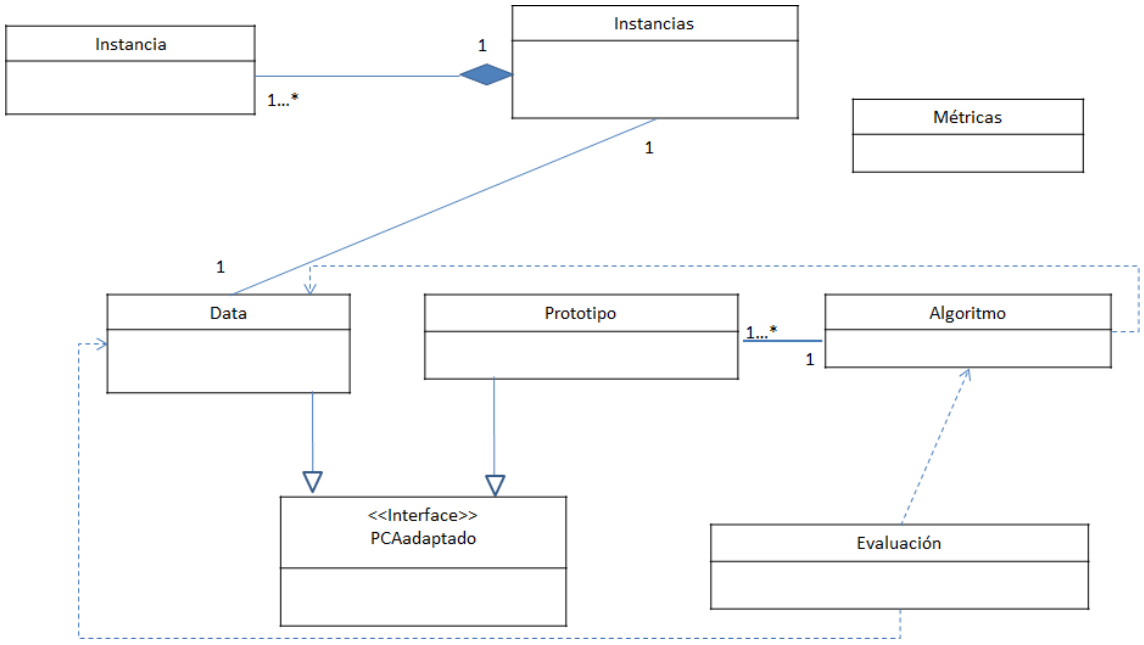


Figura 4.4. Diagrama de clases del proceso del sistema.

4.2. Experimentos del algoritmo híbrido

Dado que la propuesta diseñada puede resolver tres problemas principales como lo son clasificación, agrupamiento y el caso híbrido (planteado en el ítem 3.2.2.1), se realizaron pruebas de cada uno ellos por separado. A continuación se muestran los resultados obtenidos.

4.2.1. Clasificación

Para validar que el proceso de clasificación se esté realizando correctamente se seleccionó una variedad de problemas de MD. Las bases de datos asociadas a dichos problemas vienen dadas en la Tabla 4.1. En dicha tabla se muestra la cantidad de ejemplos de cada una de ellas, el número de atributos y sus características más resaltantes.

Tabla 4.1. Bases de datos para realizar clasificación (tomadas de Bache, K. and Lichman, M., 2013).

Base de Datos	Tamaño	Atributos	Características
iris	150	4	Tiene 3 clases con 50 instancias cada una, cada clase está asociada a un tipo de planta
diabetes	768	8	Tiene 2 clases, una como diagnosticado positivo con 268 instancias, y otra como diagnosticado negativo con 500 instancias
movement_ libra	360	91	Tiene 15 clases con 24 instancias cada una. Cada clase referencia a un movimiento de manos que es usado para el lenguaje de señas
ios	351	34	2 clases, frecuencias recolectadas por un radar para estudiar la transmisión de señales en la ionosfera
segment_challenge	1500	20	Cuenta con 7 clases, es una base de datos de imágenes segmentadas para realizar clasificación.
glass	214	10	7 clases que denotan un tipo de lente
isolet	6238	617	26 clases con 240 instancias cada clase

Para todas las bases de datos dadas en la Tabla 4.1 se crearon dos archivos, uno que contiene los datos de entrenamiento con el 66% del tamaño total y el resto para datos de prueba (34%). La métrica de evaluación tomada para validar es la precisión, ya que es la común entre los algoritmos a comparar.

Debido a que a la propuesta se le pueden modificar diferentes parámetros, en este caso se prueba con tres métricas diferentes para obtener la similaridad con el centroide, como lo son: similaridad con el coseno, distancia euclideana y correlación. El cambio de este parámetro se estudia con la finalidad de conocer el efecto que tiene el cambio del mismo sobre los resultados de la clasificación.

En la Tabla 4.2 se muestran los resultados obtenidos con nuestra propuesta y con otros algoritmos clásicos de clasificación, usando como métrica la

precisión sobre el mismo conjunto de prueba. Dichos algoritmos fueron probados con Weka (Mark Hall et al., 2009) y con la librería de Python *scikit-learn* (Pedregosa et al. 2011*),* a su vez, fueron evaluados en las mismas condiciones que en nuestra propuesta.

Los algoritmos clásicos utilizados están basados en árboles (decision tree, J48 y RandomTree), en reglas (OneR), en redes neuronales (RBF y MultilayerPerceptron), en análisis estadísticos (naiveBayes, logistic y gaussian naive bayes) y en distancia (1-NN y Nearest Shrunken Centroid).

En la Tabla 4.2 se resaltan en rojo la precisión más elevada y en negrita la segunda más elevada. Si bien es cierto que las mayores precisiones fueron obtenidas con otros algoritmos, hay que tomar en cuenta que cada algoritmo funciona mejor con cierto tipo de base de datos que con otras por su diseño propio. Sin embargo, el desempeño de nuestra propuesta es bueno en todas las bases de datos, y está a la par de los algoritmos que lograron una mayor precisión.

Al modificar la métrica de similaridad con el centroide en el modelo se obtuvieron los mismos resultados (véase la Tabla 4.2), esto debido a que el mayor peso lo tiene 1-NN a la hora de tomar la decisión final de clasificar un nuevo ejemplo en la etapa de funcionamiento. El cambio de métrica afecta principalmente al entrenamiento del modelo, y su selección depende de las características de la base de datos.

Tabla 4.2. Precisión obtenida con la propuesta y otros algoritmos clásicos de clasificación.

Algoritmo	iris	ios	segment	diabetes	glass	movement_ libra	isolet
Propuesta (euclidean)	0.9803	0.7647	0.9529	0.7279	**0.6769**	0.945	0,8710
Propuesta (cos)	0.9803	0.7647	0.9529	0.7279	**0.6769**	0.945	0,8710
Propuesta (correlation)	0.9803	0.7647	0.9529	0.7279	**0.6769**	0.945	0,8710
OneR	0.942	0.793	0.647	0.7390	0.5	0.647	-
J48	0.942	0.851	**0.954**	0.7560	0.6030	**0.954**	-
Randomtree	**0.981**	0.803	0.947	0.7410	**0.6780**	0.947	-
naiveBayes	**0.981**	0.83	0.83	0.7670	0.5890	0.83	-
logistic	**1**	0.798	0.944	**0.7950**	0.5570	0.944	-
RBF	0.965	**0.914**	0.889	**0.7980**	0.5450	0.889	-
MultilayerPe rceptron	0.965	0.836	**0.958**	0.7560	0.5850	**0.958**	-
1-NN	0.980	0.8319	0.9313	0.7241	0.6250	0.9353	**0.9014**
Nearest Shrunken Centroid	0.941	0.7226	0.8254	0.7203	0.4027	0.8235	**0.8867**
decision tree	0.961	**0.9327**	0.9294	0.6513	0.6667	0.9353	0.7957
gaussian naive bayes	0.980	0.8991	0.7491	0.7583	0.4583	0.7922	0.8028

A modo de comparación, en la Tabla 4.3 se calcula el error relativo entre el mejor resultado obtenido en la Tabla 4.2 y el resultado obtenido por nuestra propuesta. El error relativo se calcula por medio de la ec. (4.1).

$$error\ relativo = \frac{valor\ medido - valor\ real}{valor\ real}\ x100 \qquad (4.1)$$

Siendo el *valor real* la precisión más alta y el *valor medido* la precisión obtenida con nuestra propuesta.

En la Tabla 4.3 se puede observar que la mayoría de los errores relativos son pequeños, evidenciando que existe una pequeña diferencia entre el resultado que se obtuvo con nuestra propuesta y el resultado que obtuvo por el algoritmo con la precisión más elevada.

La discrepancia de los resultados obtenidos para la base de datos *ios,* está relacionada con las características de los datos y su estructura. Como se mencionó anteriormente, algunos algoritmos son mejores con ciertos tipos de bases de datos como consecuencia de su diseño propio. Al analizar la base de datos *ios,* se observa que algunos ejemplos de una clase se encuentran solapados con los ejemplos de la otra clase. Este problema suele ocurrir cuando los atributos están muy correlacionados entre sí, y los algoritmos basados en distancia no resuelven, por si solos, el problema de correlación o solapamiento. Si observamos la Tabla 4.2. los algoritmos basados en distancia (1-NN, Nearest Shrunken Centroid y nuestra Propuesta) no obtienen los mejores resultados, mientras que otros algoritmos que discriminan mejor el comportamiento de cada atributo (como los algoritmos basados en árboles) presentan un mejor desempeño. Caso similar ocurre con la base de datos *diabetes* (ver tabla 4.3).

Tabla 4.3. Error relativo entre la precisión más alta y la precisión obtenida por nuestra propuesta.

Base de datos	Error Real	Error medido	Error relativo (%)
iris	1	0.9803	1.97
ios	0.9327	0.7647	18.01
segment	0.958	0.9529	0.5324
diabetes	0.7980	0.7279	8.784
glass	0.6780	0.6769	0.1622
movement_libra	0.958	0.945	1.3570
isolet	0.9014	0.8710	3.372

4.2.2. Semi-supervisados

Los algoritmos semi-supervisados son especializados para resolver o mejorar los resultados de clasificación, y son recomendados cuando se tiene gran cantidad de datos. El problema es el mismo de clasificación, la única diferencia es que se entrena el modelo de manera diferente, denominada "co-training" (sección 2.4).

Esta prueba consiste en entrenar el modelo con un pequeño porcentaje de ejemplos etiquetados, generando así un modelo con un número finito de clases, seguidamente se evalúa el resto de ejemplos (removiendo su etiqueta) sobre el primer modelo, y se etiquetan según la clase a la que fueron asignadas. El modelo de clasificación resultante viene dado por el resultado de las dos etapas descritas.

Para este tipo de entrenamiento, su efectividad se evalúa en la segunda etapa. El proceso de evaluación consiste en comparar las etiquetas originales del conjunto utilizado en la segunda fase, con el resultado obtenido después de haber sido etiquetados por el modelo. Dicha evaluación se realiza con el cálculo del error.

En este caso, se prueba nuestro algoritmo con los benchmarks dados en la Tabla 4.4, los cuales son problemas que los algoritmos clásicos de clasificación no son capaces de resolver.

Tabla 4.4. Bases de datos para realizar clasificación por medio de aprendizaje semisupervisado (Chapelle, O et al. 2006).

Nombre	Tamaño	Atributos	Clases	Características
g241c	1500	241	2	Base de datos artificial
g241d	1500	241	2	Base de datos artificial
Digit1	1500	241	2	Base de datos artificial
USPS	1500	241	2	Imágenes de números escritos a mano con ruido, los números 2 y 5 pertenecen a una clase y el resto de los números a la otra clase
COIL	1500	241	6	Imágenes de 6 objetos tomadas de diferentes ángulos y con ruido
BCI	400	117	2	Señales enviadas al cerebro al mover la mano a la derecha o a la izquierda, sensado con electrodos.

En la Figura 4.5 se observa gráficamente como están distribuidos los ejemplos de las bases de datos *g241c* y *g241d*. Se observa que existe gran complejidad a la hora de definir las fronteras de las clases. Estos ejemplos fueron construidos por Chapelle y sus colaboradores, como casos extremos para la validación de algoritmos especializados semi-supervisados.

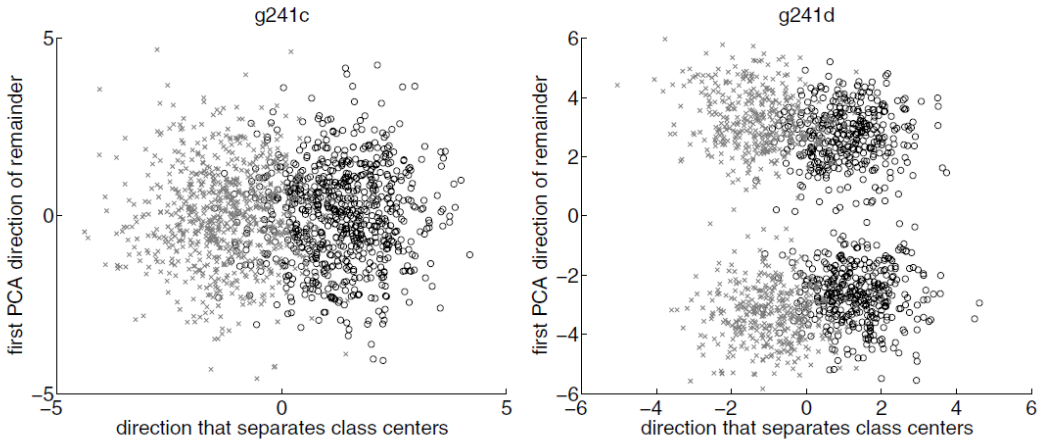


Figura 4.5. (a) PCA de la base de datos g241c (b) PCA de la base de datos g241c. Tomado de Chapelle, O et al. 2006.

Digit1 es una base de datos artificial de imágenes que representan la escritura del número 1 en diferentes ángulos y con ruido agregado. En la figura 4.6 se puede observar una de las imágenes que contiene la base de datos.

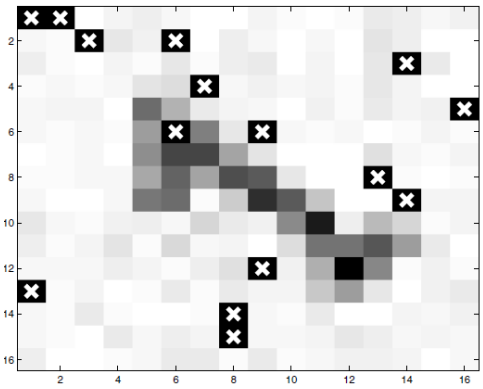


Figura 4.6. Imagen de un ejemplo de la base de datos Digit1. Tomado de Chapelle, O et al. 2006.

El resultado generado por la propuesta se compara con diferentes algoritmos especializados semi-supervisados (Chapelle, O et al. 2006). Los algoritmos semi-supervisados dados son: basados en la suposición de manifold (SVM, Discrete Reg, QC, Laplacian RLS, CHM, LEM, SGT) y basados en la suposición de cluster y suavidad (TSVM, Data-Dep Reg, Cluster-Kernel y LDS). Además, se compara con 1-NN, ya que el mismo es utilizado por otros algoritmos para el fortalecimiento de sus algoritmos, al igual que la presente prepuesta.

En la Tabla 4.5 se muestran los resultados obtenidos por nuestra propuesta, mientras que en la Figura 4.7 se observan los resultados obtenidos por Chapelle y sus colaboradores en sus pruebas. Cabe destacar que las pruebas están hechas bajo las mismas condiciones, y con las mismas particiones de datos de entrenamiento para la primera etapa y para la segunda de "co-trainning".

Es claro que algunos algoritmos funcionan mejor con algunas bases de datos que con otras. Además, los algoritmos de la Figura 4.7 son algoritmos semi-supervisados destinados solo a problemas de clasificación con grandes cantidades de datos. Los resultados obtenidos por nuestra propuesta deberían ser semejantes a los resultados obtenidos con 1-NN (debido al diseño propio

del algoritmo), se pudo observar en Tabla 4.6 que los resultados son cercanos, y que la propuesta puede soportar problemas de clasificación con estas características.

La diferencia encontrada entre los resultados de nuestra propuesta y 1-NN, se debe al diseño propio de nuestro algoritmo. En algunas bases de datos; nuestra propuesta mejora los resultados dados por 1-NN (*Digit1, USPS y BC1*), mientras que en otras empeora los resultados (*g241c*, *g241d y COIL*). Esto depende directamente de la estructura de los datos y de los criterios que se han usado en nuestra propuesta, además de un factor adicional que no evalúa 1-NN, como lo es la similaridad con el centroide.

Las bases de datos artificiales *g241c* y *g241d* son problemas de clasificación complejos de resolver por nuestra propuesta, debido que no cumple con los criterios que hemos planteado para la creación de prototipos (ver sección 3.2.2.1, ítems ii, iii y vi). Recordando que en la primera etapa del co-trainning se crean prototipos por medio de aprendizaje supervisado, después se evalúan los datos no etiquetados utilizando los criterios del aprendizaje no supervisado. Si observamos la Figura 4.5, no existe una separación evidente de las clases, es por ello que se obtienen errores de prueba elevados. Por otro lado, con la base de datos *BC1* se obtuvieron errores elevados con todos los algoritmos, siendo el algoritmo *Laplacian RLS* con el menor error.

Tabla 4.5. Errores de prueba (%) obtenidos con la propuesta, con 100 ejemplos etiquetados del total de ejemplos.

	g241c	g241d	Digit1	USPS	COIL	BC1
Propuesta	44	45	3.71	5.78	21.35	42.6

		g241c	g241d	Digit1	USPS	COIL	BCI
	1-NN	43.93	42.45	3.89	5.81	17.35	48.67
	SVM	23.11	24.64	5.53	9.75	22.93	34.31
21.2.8	MVU + 1-NN	43.01	38.20	2.83	6.50	28.71	47.89
21.2.8	LEM + 1-NN	40.28	37.49	6.12	7.64	23.27	44.83
21.2.4	QC + CMN	22.05	28.20	3.15	6.36	10.03	46.22
21.2.6	Discrete Reg.	43.65	41.65	2.77	4.68	9.61	47.67
21.2.1	TSVM	18.46	22.42	6.15	9.77	25.80	33.25
21.2.1	SGT	17.41	9.11	2.61	6.80	–	45.03
21.2.10	Cluster-Kernel	13.49	4.95	3.79	9.68	21.99	35.17
21.2.3	Data-Dep. Reg.	20.31	32.82	2.44	5.10	11.46	47.47
21.2.11	LDS	18.04	23.74	3.46	4.96	13.72	43.97
21.2.5	Laplacian RLS	24.36	26.46	2.92	4.68	*11.92*	31.36
21.2.7	CHM (normed)	24.82	25.67	3.79	7.65	–	36.03

Figura 4.7. Error de prueba (%) con 100 ejemplos etiquetados del total (Chapelle, O et al. 2006).

4.2.3. Agrupamiento

Para la validación de las agrupaciones que realiza la propuesta, es necesario realizar un conjunto de pruebas sobre grupos de diferentes tamaños y formas, es por ello que se plantean las bases de datos dadas en la Tabla 4.6. Las mismas cumplen con una variedad de características que se podrían presentar en cualquier problema de agrupamiento, y son benchmarks elementales para la validación de algoritmos de agrupamiento.

Tabla 4.6. Bases de datos para realizar agrupamiento tomadas de Ultsch, A. 2005 y Pedregosa et al. 2011.

Base de Datos	Tamaño	Atributos	Características
Hepta	212	3	7 clusters bien definidos
Lsun	400	2	3 clusters con diferentes varianzas
wingNut	1016	2	2 clusters con variación de distancia y densidad
Target	770	2	6 cluster con valores atípicos
chainLink	1000	3	2 clusters linealmente no separables
No estructure	1500	2	1 cluster sin estructura alguna
noisyMoons	1500	2	2 clusters en forma de media luna
noisyCircles	1500	2	1 cluster en forma de circulo que contiene otro cluster en forma de circulo

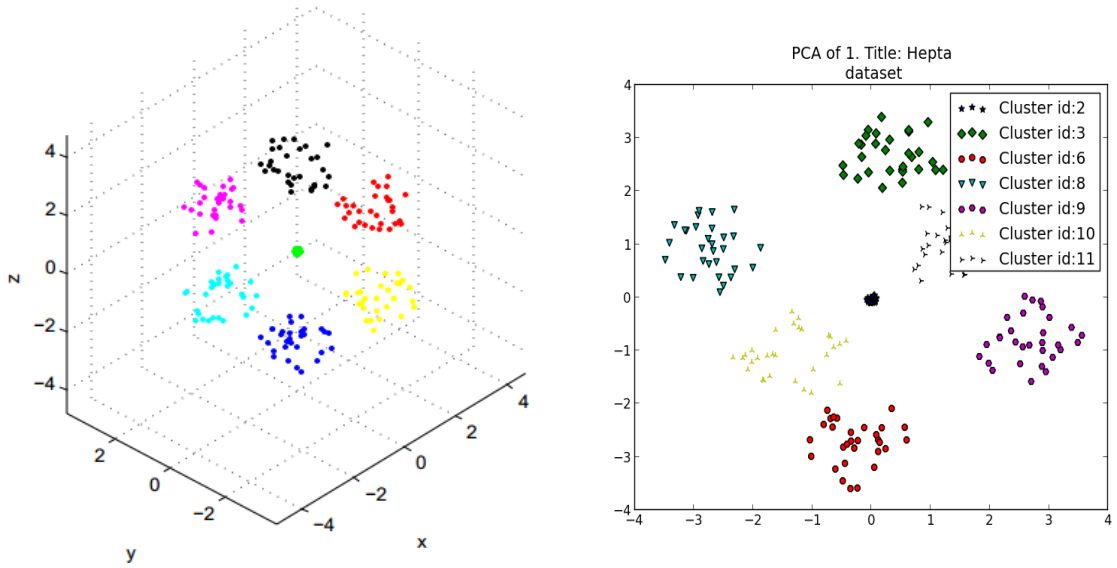
Continua en la próxima página …

			Viene de la página anterior …
Blobs	1500	2	3 manchas de Gauss isotrópicas para la agrupación
Tetra	400	3	4 clusters muy cercanos entre si
engyTime	4096	2	2 clusters, es un problema de mezcla gaussiana entre los clusters

Cabe destacar que las métricas de varianza y distancia promedio afectan en gran medida al desarrollo adecuado de los algoritmos de agrupamiento en general. En nuestro caso es vital, ya que como se presentó en el Capítulo 3, se debe definir a priori el umbral de vecindad.

Por defecto, el umbral de vecindad puede ser tomado como la varianza, la desviación estándar o la distancia promedio entre las instancias. Esta suposición es válida cuando se tienen grupos bien definidos, grupos muy cercanos o con varianzas iguales. En caso de contar con diferentes varianzas, es recomendable comenzar con un valor pequeño de umbral, y el algoritmo se va adaptando, calculando las distancias promedios entre los ejemplos hasta conseguir los umbrales adecuados para cada grupo.

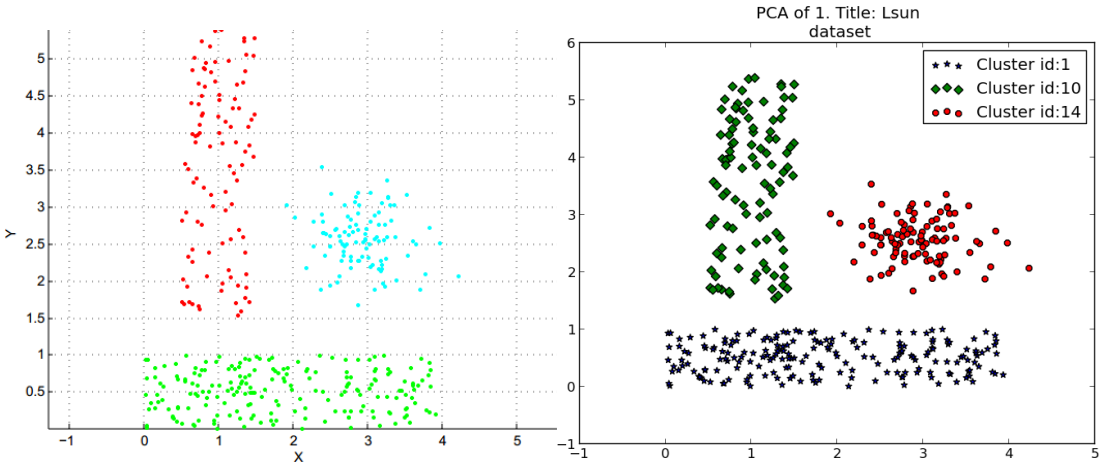
En la Figura 4.8-(a) se presenta el primer ejemplo, el cual tiene grupos bien definidos con varianzas iguales. En este caso el problema es trivial, el resultado dado por nuestra propuesta se muestra en la Figura 4.8-(b). Se inicializó el umbral de vecindad con desviación estándar de todos los datos ρ.



(a) (b)

Figura 4.8. (a)Datos originales de la base de datos Hepta(b) Resultado generado por la propuesta con la base de datos Hepta con $\boldsymbol{uv = \rho}$ como valor inicial.

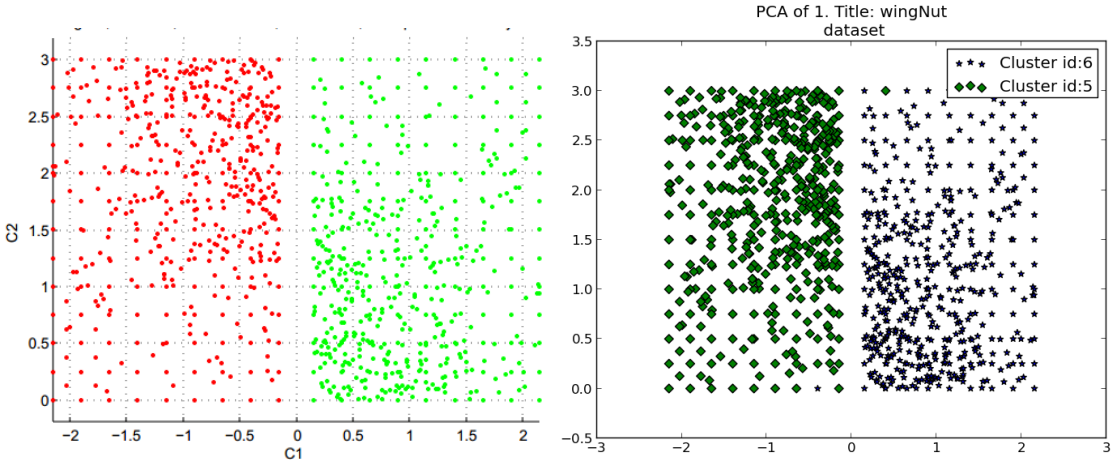
El segundo problema dado en la Figura 4.9-(a) presenta un problema más complejo, con tres grupos de diferentes formas y varianzas. Sin embargo, nuestro algoritmo puede manejar grupos de diferentes estructuras, como se muestra en la Figura 4.9-(b). El umbral de vecindad inicial utilizado para dicho problema para todos los grupos es $uv = {}^{\rho}/_{2}$, siendo ρ la desviación estándar de todos los datos.



(a) (b)

Figura 4.9. (a)Datos originales de la base de datos Lsun(b) Resultado generado por la propuesta con la base de datos Lsun.

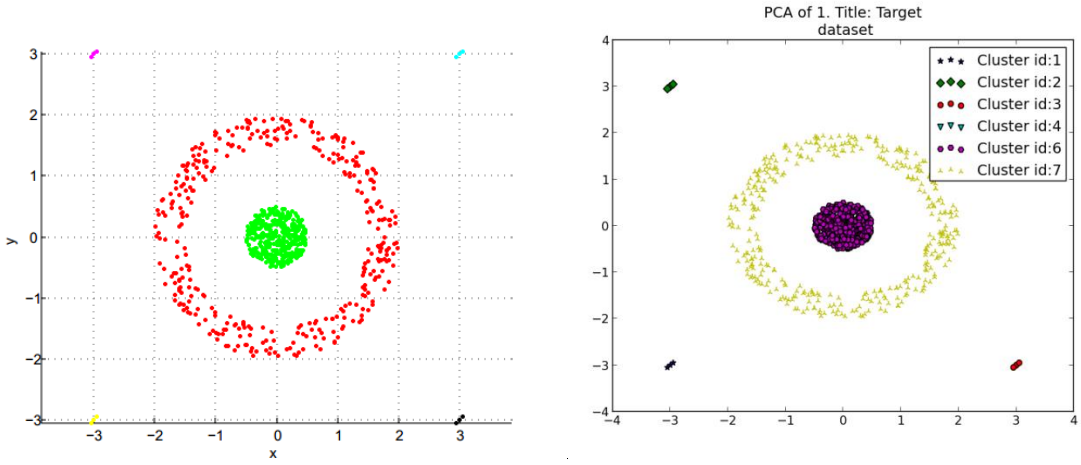
El ejemplo de la Figura 4.10-(a) combina el problema de distancias diferentes entre las instancias y el problema de diferentes densidades a lo largo de la zona comprendida por cada grupo. En este caso, el umbral de vecindad es la varianza y el resultado obtenido es dado en la Figura 4.10-(b).



(a) (b)

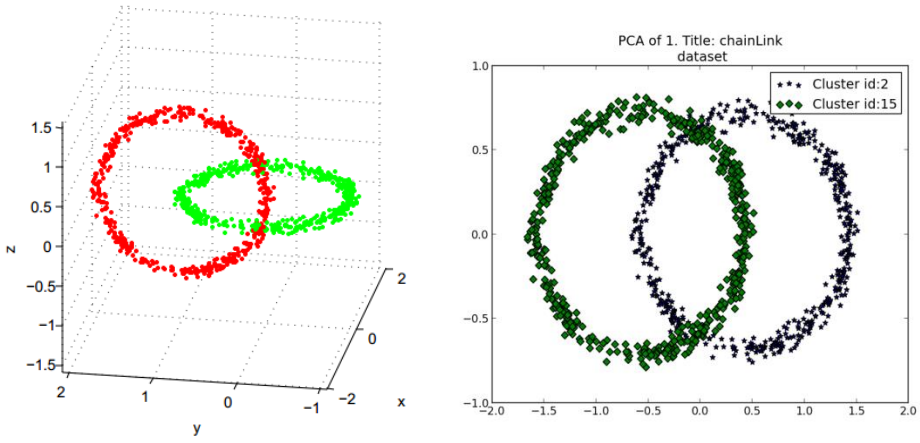
Figura 4.10. (a)Datos originales de la base de datos WingNut(b) Resultado generado por la propuesta con la base de datos WingNut.

La Figura 4.11 y 4.12 muestran otros problemas de agrupamiento que el algoritmo presentado puede resolver. Estos dos problemas resaltan el hecho de que la similaridad con el centroide, para tomar la decisión de asignar un ejemplo a un grupo, no es adecuada; debido a que la forma de los grupos no permite que el centroide sea una medida representativa de cada grupo. Este problema lo puede resolver la propuesta por el planteamiento del vecino más cercano, ya que se van agrupando los ejemplos vecinos hasta formar los anillos de las Figuras 4.11 y 4.12.



(a) (b)

Figura 4.11. (a)Datos originales de la base de datos Target (b) Resultado generado por la propuesta con la base de datos Target.



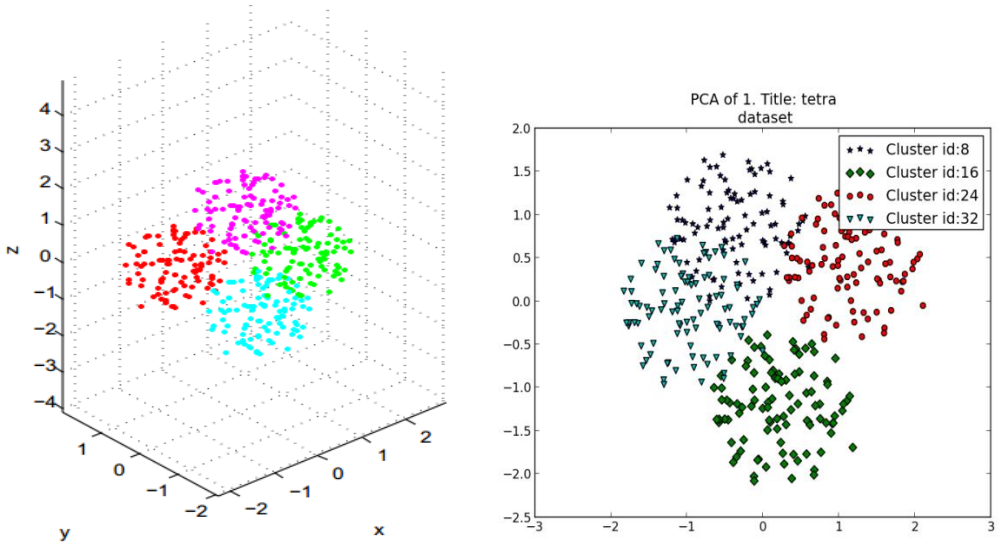
(a) (b)

Figura 4.12. (a)Datos originales de la base de datos ChainLink (b) Resultado generado por la propuesta con la base de datos ChainLink.

Se realizó otra prueba para la validación de agrupamiento, utilizando una medida externa como lo es la precisión. Normalmente, los algoritmos de agrupamiento no utilizan las métricas externas para la evaluación de sus algoritmos, debido a que son muy costosas computacionalmente y porque no se cuentan con las etiquetas de los ejemplos.

Para realizar esta prueba se utilizaron dos bases de datos, las cuales tienen ejemplos con sus etiquetas, las mismas son: *Tetra* y *engyTime*.

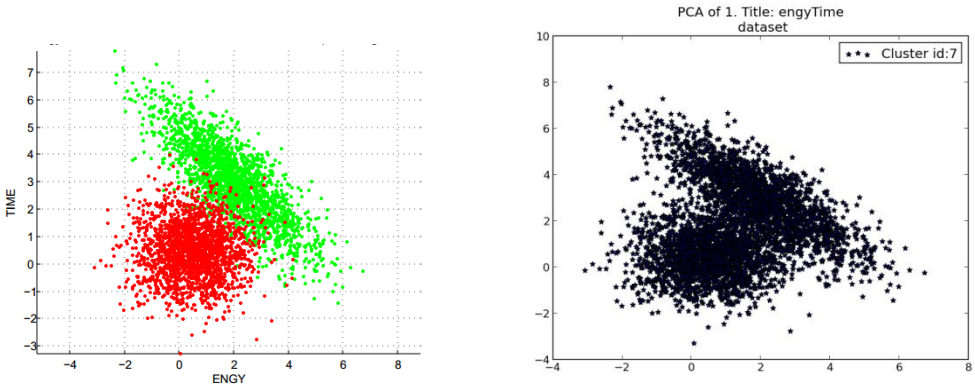
En la Figura 4.13 se observa el resultado obtenido al evaluar la base de datos tetra en el algoritmo, la precisión obtenida es *p=1*. Si bien es cierto, que los grupos están muy cercanos, el algoritmo es capaz de crear los 4 grupos sin errores.



(a) (b)

Figura 4.13. (a)Datos originales de la base de datos Tetra (b) Resultado generado por la propuesta con la base de datos Tetra.

El caso de la base de datos *engyTime,* dado en la Figura 4.14, es más complejo que los estudiados previamente, ya que se cuenta con dos grupos solapados. Este caso no puede ser resuelto por el algoritmo propuesto, debido a que una de las suposiciones de la propuesta se basa en que dos grupos están separados por una distancia "grande" o por baja densidad en sus fronteras. Esta suposición no se cumple para este ejemplo, por lo tanto el resultado dado en la Figura 4.14 evidencia que el algoritmo no diferencia dos grupos en el conjunto de entrenamiento y crea uno solo sin distinción.



(a) (b)

Figura 4.14. (a)Datos originales de la base de datos EngyTime (b) Resultado generado por la propuesta con la base de datos EngyTime.

Como prueba adicional, se evalúan los agrupamientos realizados previamente con medidas internas (véase la Tabla 4.7), como lo son la cohesión y la separación promedio, simplemente para dar una idea de que tan bien están separados y agrupados los grupos creados. Se utiliza la métrica propuesta en la sección 3.2.3 ec.(3.7), la misma aporta información acerca del rendimiento del algoritmo para la creación de grupos.

En la Tabla 4.7 se puede observar que los resultados obtenidos para las bases de datos *Hepta*, *Target* y *Tetra* son cercanos a 1, lo cual indica que se crearon grupos bien definidos.

El resultado obtenido para *Lsun* y *wingNut* indica que no es concluyente; sin embargo, al observar gráficamente los resultados (Figuras 4.9 y 4.10) se puede justificar que los grupos fueron creados adecuadamente, y que la métrica no es concluyente debido a las características propias de los datos. *Lsun* está compuesto de grupos de diferentes formas y longitudes, esto afecta al cálculo de la cohesión y la separación. Por otro lado, en los grupos creados con la base de datos *wingNut* no existe una alta cohesión entre las instancias, debido a que las mismas están muy dispersas y la distancia entre ambos grupos es muy pequeña.

En el caso de la base de datos *chainLink* la forma de los grupos no permite que esta métrica sea representativa, ya que son dos anillos que se enlazan en un espacio tridimensional. Los centroides de ambos anillos están ubicados de forma tal que no es posible medir la concentración de ejemplos de los grupos por medio de la cohesión. Mientras que la separación tampoco se puede determinar porque ambos grupos están enlazados. Esta estructura genera confusión para el cálculo de las métricas de evaluación del agrupamiento.

En la base de datos *engyTime* las métricas de evaluación no aplican porque se creó un solo grupo.

Tabla 4.7. Métricas para la evaluación de los agrupamientos utilizando medidas externas.

Base de Datos	Co_μ	Sep_μ	$M = 1-Co_\mu/Sep_\mu$
Hepta	0.6447	3.5417	0.818
Lsun	0.8699	1.6868	0.4656
wingNut	0.8982	1.0104	0.1111
Target	0.2197	4.5073	0.95126
chainLink	0.9956	0.5002	-0.9904
Tetra	0.2468	1.5504	0.8409
engyTime	2.0514	-	-

Para comparar la propuesta con otros algoritmos de agrupamiento, se utilizan las bases de datos *noisyCicles*, *noisyMoons*, *blobs* y *No estructure (*véase la Tabla 4.6). Los grupos originales de estas bases de datos se muestran en la Figura 4.15. Seguidamente se evalúan las bases de datos en siete algoritmos de agrupamiento (Pedregosa et al. 2011): basados en métodos jerárquicos (Ward y AgglomerativeClustering), basado en densidad (DBSCAN), basados en prototipos (MiniBatchKMeans y MeanShift), y otros métodos (AffinityPropagation y SpectralClustering).

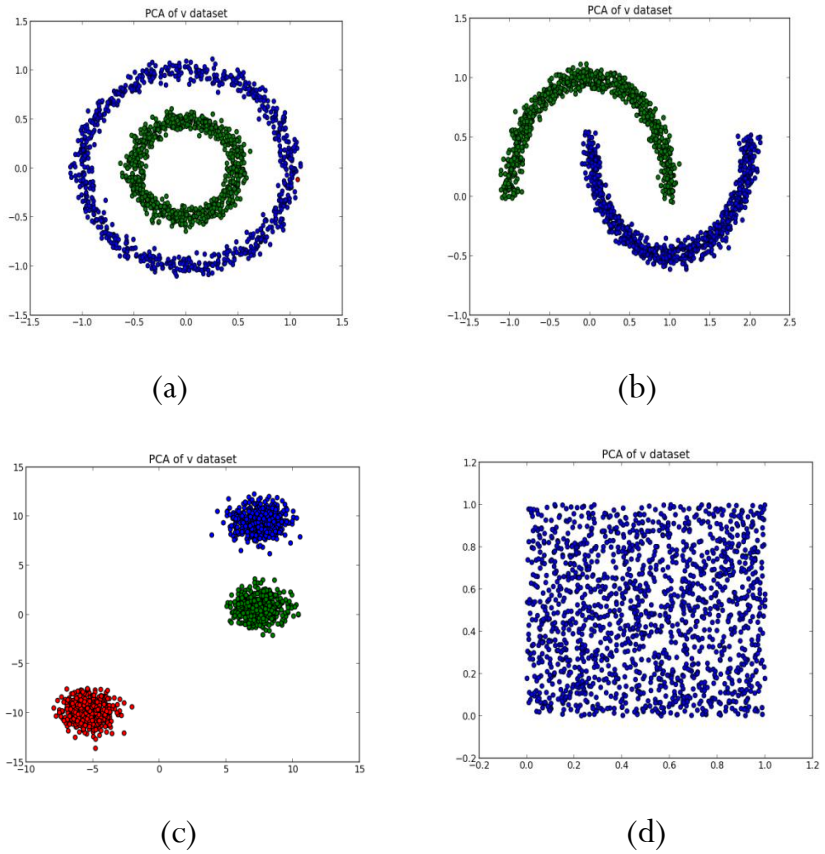
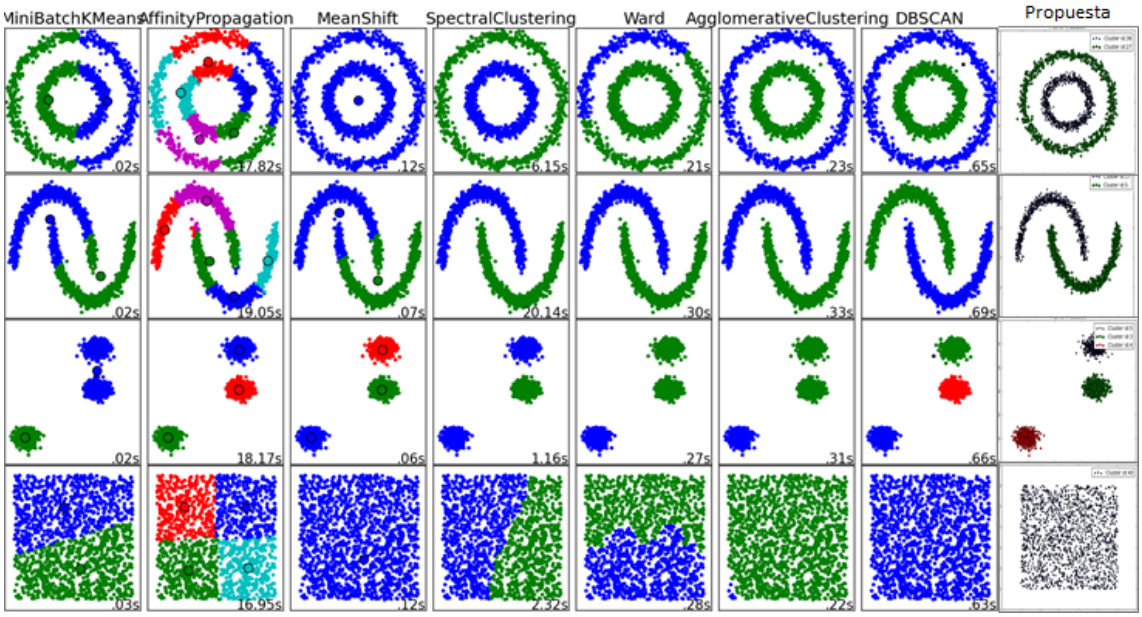


Figura 4.15. (a) Datos originales de la base de datos *NoisyCircles* (b Datos originales de la base de datos *NoisyMoons* (c) Datos originales de la base de datos *Blobs* (d) Datos originales de la base de datos *No estructure*.

En la Figura 4.16 se pueden observar los resultados gráficamente de los algoritmos y de la propuesta presentada. En la primera fila se utilizó la base de datos *noisyCicles*, en la segunda *noisyMoons*, en la tercera *blobs,* y por último *No estructure* (véase la Tabla 4.7). Los resultados obtenidos con la propuesta son satisfactorios, están al nivel de los algoritmos clásicos y especializados de agrupamiento, ya que genera los grupos originales (véase la Figura 4.15) para cada una de las bases de datos.



(a) (b) (c) (d) (e) (f) (g) (h)

Figura 4.16. (a) Resultados obtenidos con MiniBatchKMeans (b) Resultados obtenidos con AffinityPropagation. (c) Resultados obtenidos con MeanShift. (d) Resultados obtenidos con SpectralClustering (e) Resultados obtenidos con Ward. (f) Resultados obtenidos con AgglomerativeClustering. (g) Resultados obtenidos con DBSCAN. (h) Resultados obtenidos con la propuesta.

4.2.4. Híbrido

En la literatura no se consiguen ejemplos híbridos como los que se plantean resolver con nuestra propuesta, ya que la mayoría de las bases de datos son procesadas de manera que este tipo de casos no ocurra. Debido a esto, se construye un conjunto de bases de datos artificiales para demostrar el funcionamiento del algoritmo en el caso que hemos denominado híbrido, dado en el ítem 3.2.2.1 del Capítulo 3.

Inicialmente se había definido que un caso híbrido ocurre cuando en los datos de entrenamiento o de prueba, se pueden conseguir las tareas de clasificación y agrupamiento en conjunto. Esto ocurre cuando tenemos conjuntos de ejemplos etiquetados y conjuntos de ejemplos sin etiqueta en la misma base de datos, es por ello que se plantea la creación de una serie de base de datos que simule estas condiciones.

4.2.4.1. Experimento híbrido 1

Para la creación de la base de datos base se utilizaron mezclas gaussianas (Pedregosa et al. 2011), en las cuales se definen la cantidad de ejemplos a crear, atributos, clases y la varianza entre los ejemplos. La base de datos original está compuesta por cinco clases: tres bien definidas (clases 2, 3 y 4) y dos solapadas (clases 0 y 1), como se muestra en la Figura 4.17.

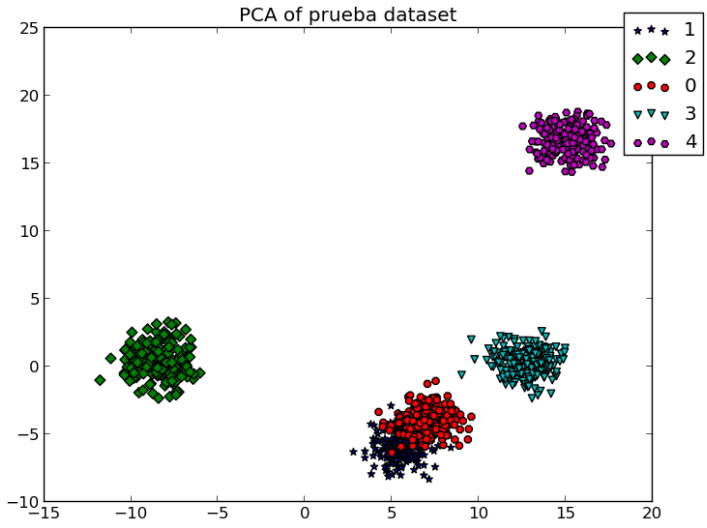


Figura 4.17. Datos artificiales originales del experimento híbrido 1.

Para realizar las pruebas híbridas artificiales, se plantean los siguientes escenarios que pueden ocurrir al utilizar el algoritmo propuesto:

- Entrenar el modelo con todo el conjunto de datos sin contar con ejemplos de prueba; además, este conjunto de datos de entrenamiento está formado por ejemplos etiquetados y no etiquetados.
- Entrenar el modelo con 66% de los ejemplos de la base de datos y el resto (34%) es utilizado como datos de prueba. Al igual que el caso anterior, ambos conjuntos contienen ejemplos etiquetados y no etiquetados.

- Entrenar el modelo solo con ejemplos etiquetados, y probar el modelo (funcionamiento en línea) con ejemplos no etiquetados que no pertenezcan a ninguno de los prototipos creados en la etapa de entrenamiento, esto con la finalidad de demostrar la capacidad del algoritmo para crear nuevos grupos.

En la Tabla 4.8 se muestran los cambios realizados sobre la base de datos artificial original, y en qué condiciones se probará el algoritmo. Siendo *N° Grupos* la cantidad de grupos que contiene la base de datos, *N° Clases* la cantidad de clases que contiene la base de datos, *Entrenamiento* son los ejemplos usados para el entrenamiento del modelo, y *Prueba* son los ejemplos usados para prueba del modelo.

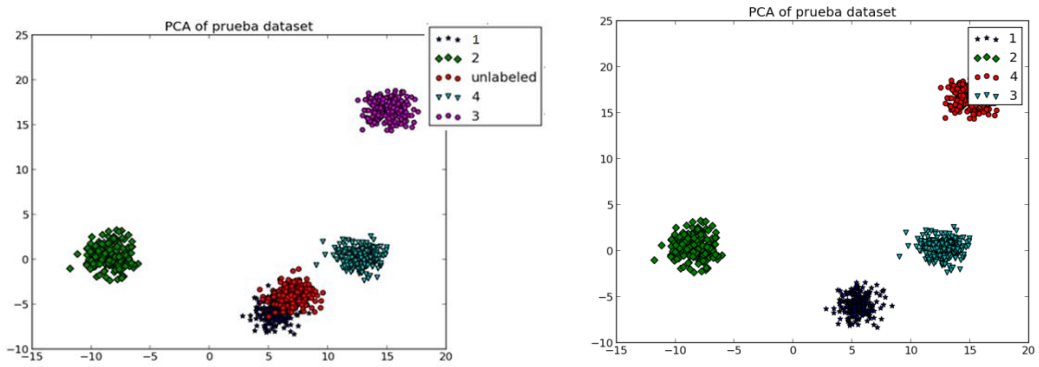
Tabla 4.8. Resumen de las características de las bases de datos artificiales creadas para la prueba 1.

Base de datos	N° Grupos	N° Clases	Entrenamiento	Prueba
Artificial1	1 (etiqueta 0)	4	100%	-
Artificial2	1 (etiqueta 0)	4	66%	34%
Artificial3	1 (etiqueta 2)	4	100%	-
Artificial4	1 (etiqueta 2)	4	66%	34%
Artificial5	2 (etiqueta 1 y 3)	3	66%	34%
Artificial6	1 (etiqueta 2)	4	Solo etiquetados	Solo no etiquetados

a. Prueba con *Artificial1* y *Artificial2*

Para la prueba con *Artificial1* se remueve la etiqueta de la clase 0 (véase la Figura 4.18-(a)) y se entrena el algoritmo con el conjunto completo de ejemplos de la base de datos artificial. El resultado es dado en la Figura 4.18-(b), el cual evidencia que se crea una sola clase etiquetando los ejemplos de la clase 0 con la clase etiqueta de la clase 1. Es evidente que el algoritmo a la hora de realizar el entrenamiento del modelo no detecta distinciones cuando

se encuentra en este tipo de solapamiento, y sigue la suposición de que un ejemplo no etiquetado pertenece a la clase con la más alta similaridad.



(a) (b)

Figura 4.18. (a)Datos originales eliminando la etiqueta de la clase 0(b) Resultado generado por la propuesta con la base de datos artificial sin la etiqueta de la clase 0.

La prueba con *Artificial2* consiste en dividir el conjunto *Artificial1* entre 66% de datos de entrenamiento y 34% de datos de prueba. El resultado generado es el mismo dado en la Figura 4.18-(b). Se realiza este experimento con la finalidad de demostrar que se genera el mismo resultado en ambos escenarios presentados con *Artificial1* y *Artificial2*.

b. Prueba con *Artificial3* y *Artificial4*

Se remueve la etiqueta de la clase 2 y se entrena el modelo con el conjunto completo de ejemplos en la base de datos. El resultado para *Artificial3* es dado en la Figura 4.19, el grupo 20 está bien definido, por lo tanto, el algoritmo no tiene problemas en detectarlo. Por otro lado, crea las clases correspondientes a los ejemplos etiquetados.

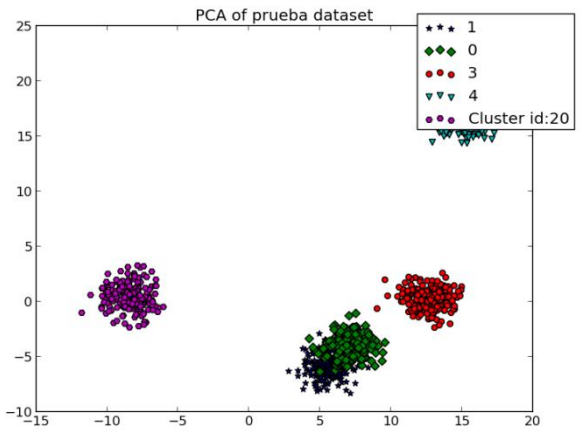


Figura 4.19. Resultado generado por la propuesta con la base de datos artificial sin la etiqueta de la clase 2.

Al dividir *Artificial3* en un conjunto de entrenamiento con 66% y un conjunto de prueba de 34%, se crea *Artificial4*. El resultado generado al entrenar el modelo es el mismo dado en la Figura 4.18.

c. Prueba con *Artificial5*

Sobre la base de datos artificial original se remueven las etiquetas de la clase 1 y 3. Seguidamente se entrena el modelo con 66% del conjunto completo de ejemplos y el resto es utilizado como ejemplos de prueba, el resultado se muestra en la Figura 4.20. Los grupos 11 y 12 son difíciles de observar en la imagen, ya que están formados por un solo elemento, y son ejemplos aislados en los cuales el algoritmo no encontró una similaridad alta con respecto a los demás grupos/clases existente, los mismos se enmarcaron en un triángulo negro. Normalmente, para algunos algoritmos de agrupamiento estos ejemplos pueden ser etiquetados como ruido y se eliminan de la base de datos, ya que se encuentran en las fronteras con otros grupos. Por otro lado, el grupo representado por la etiqueta 1 no fue detectado por el algoritmo, ya que sus ejemplos se encuentran solapados con los ejemplos de

la clase 0, y el diseño propio del algoritmo no distingue dicho solapamiento; por lo tanto, solo se crea una clase con el conjunto completo de ejemplos.

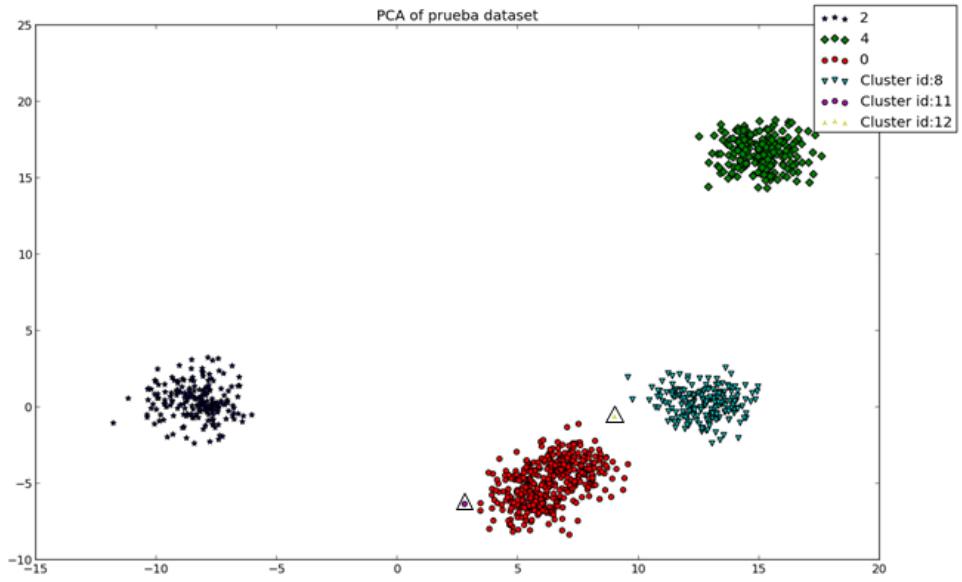


Figura 4.20. Resultado generado por la propuesta con la base de datos artificial sin la etiqueta de la clase 3 y 1.

d. Prueba con Artificial6

Este caso es muy particular, ya que se construye con la intensión de probar que nuestra propuesta es capaz de conformar grupos nuevos después de haber sido entrenada (en la etapa de funcionamiento), cosa que no es común en los algoritmos de minería de datos. Se plantea colocar en el conjunto de entrenamiento cuatro clases etiquetadas (clase 1, 3, 4 y 5 de la Figura 4.17), y en el conjunto de prueba todos los ejemplos de una clase sin etiqueta (clase 2). El resultado es dado en la Figura 4.21. Como se esperaba, crea un conjunto nuevo para estos datos que son completamente diferentes a los datos con los que se entrenó el modelo.

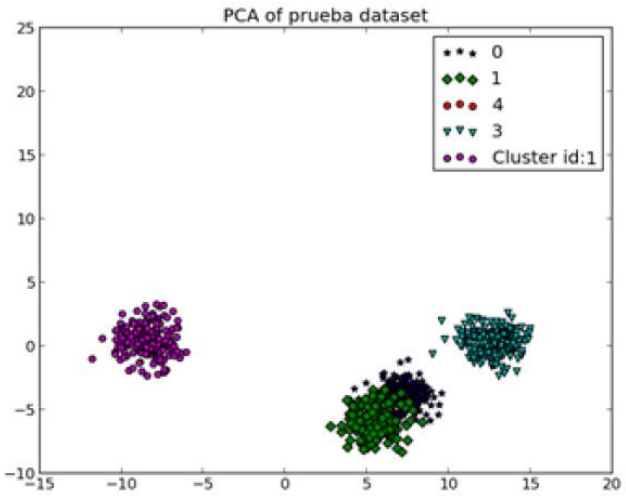


Figura 4.21. Resultado generado por la propuesta con la base de datos artificial sin la etiqueta de la clase 2.

A continuación, se presenta en la Tabla 4.9 la evaluación de las pruebas realizadas previamente con las métricas de evaluación del algoritmo. En la etapa de entrenamiento se calculó la precisión de las clases construidas, la cohesión y la separación de los grupos, y la métrica híbrida. Con todas las bases de datos artificiales se obtuvo un valor cercano a 1 en la métrica hibrida, lo que refleja una construcción adecuada de los prototipos. Para la evaluación del funcionamiento en línea se obtuvo la precisión de los ejemplos etiquetados, en el caso particular de *Artificial6* la precisión no se calcula porque los ejemplos de prueba no eran etiquetados, y no estaban relacionados con ninguna de las clases construidas por el modelo.

Tabla 4.9. Métricas para la evaluación del algoritmo de la prueba 1.

	Entrenamiento					**Funcionamiento en línea**
Base de datos	p	Co_μ	Sep_μ	1- Co_μ/Sep_μ	Métrica hibrida H	Precisión
Artificial2	0.8120	-	-	-	0.8120	0.8088
Artificial4	1	1.3228	16.0854	0.9177	0.9835	0.8926
Artificial5	0.7504	0.7795	9.8920	0.9212	0.8358	0.6872
Artificial6	1	1.3228	16.0854	0.9177	0.9835	-

4.2.4.2. Experimento híbrido 2

En este caso, las mezclas gaussianas tienen una varianza más grande y las clases están muy cercanas. La base de datos original está compuesta por cuatro clases, como se muestra en la Figura 4.22.

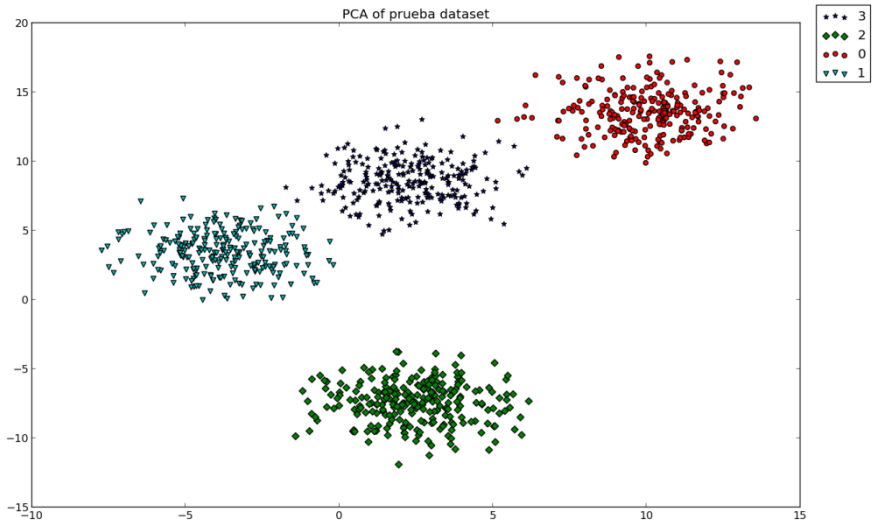


Figura 4.22. Datos artificiales originales del experimento híbrido 2

Las pruebas a realizar constan en la creación de conjuntos con etiqueta y sin etiqueta, al igual que el caso anterior. En la Tabla 4.10 se muestran los cambios realizados a la base de datos original, y en qué condiciones se va a probar el algoritmo.

Tabla 4.10. Resumen de las características de las bases de datos artificiales creadas para la prueba 2.

Base de datos	N° Grupos	N° Clases	Entrenamiento (%)	Prueba (%)
Artificial1	1 (etiqueta 0)	3	66%	34%
Artificial2	1 (etiqueta 2)	3	66%	34%
Artificial3	2 (etiqueta 1 y 3)	2	66%	34%

a. Prueba con *Artificial1*

En la prueba con *Artificial1* se remueve la etiqueta de la clase 0 (véase la Figura 4.22). El resultado es dado en la Figura 4.23, se observa que el grupo 70 es el que representa la clase 0, y los demás grupos son ejemplos de frontera que se pueden considerar como ruido.

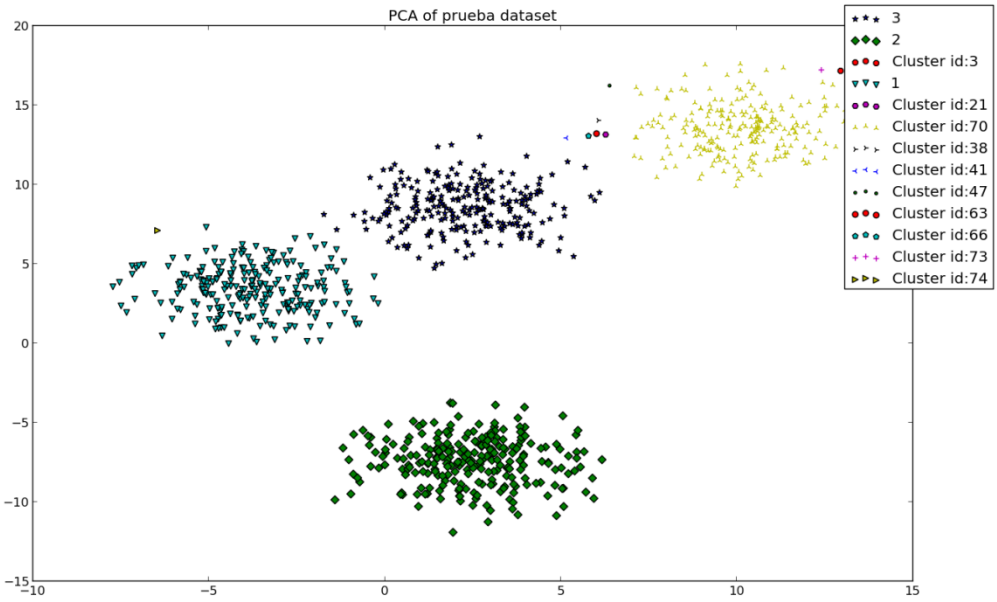


Figura 4. 23. Resultado generado por la propuesta con la base de datos artificial sin la etiqueta de la clase 0.

a. Prueba con *Artificial2*

Se remueve la etiqueta de la clase 2 (véase la Figura 4.22), el cual está formado por un conjunto de ejemplos con características bien definidas, por lo tanto no es difícil para la propuesta la creación del nuevo grupo como se muestra en la Figura 4.24.

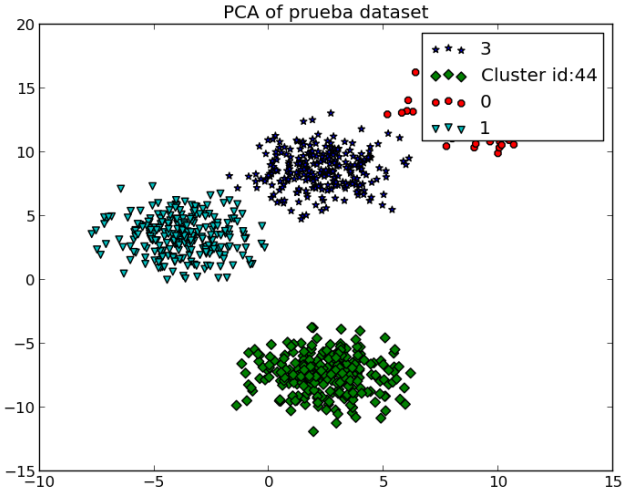


Figura 4.24. Resultado generado por la propuesta con la base de datos artificial sin la etiqueta de la clase 2.

a. Prueba con *Artificial3*

Como prueba adicional, se remueve la etiqueta de dos clases (clase 1 y 3, véase la Figura 4.22), en ambos casos el algoritmo es capaz de la creación de dos grupos (10 y 41), y al igual que previos ejemplos, crean grupos con un solo ejemplo en las fronteras, como se muestra en la Figura 4.25.

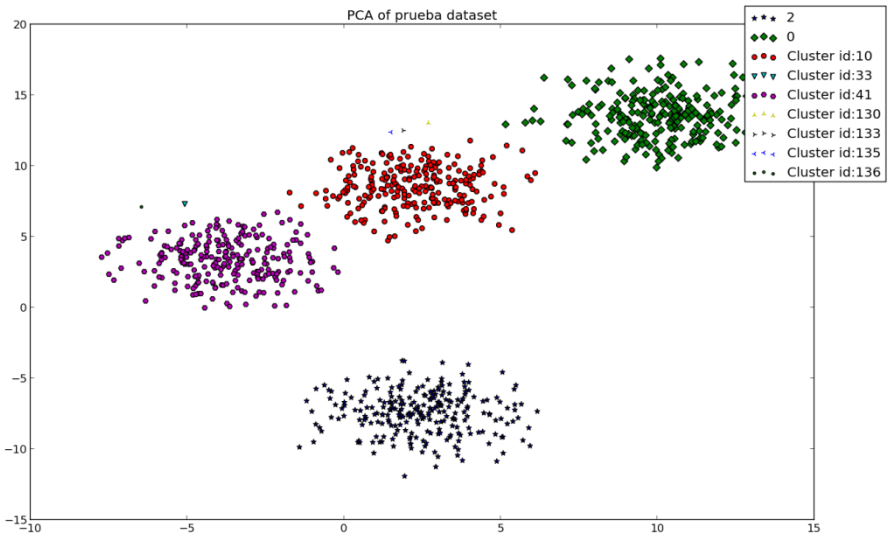


Figura 4.25. Resultado generado por la propuesta con la base de datos artificial sin la etiqueta de la clase 1 y 3.

En la Tabla 4.11 se evidencian los resultados de la evaluación de las pruebas realizadas previamente, con las métricas de evaluación del algoritmo. Se puede concluir que los resultados son satisfactorios, tanto en la fase de entrenamiento como en el funcionamiento en línea.

Tabla 4.11. Métricas para la evaluación del algoritmo de la prueba 2.

	Entrenamiento					**Funcionamiento en línea**
Base de datos	Precisión	Promedio Cohesión	Promedio Separación	1-Co/Sep	Métrica híbrida	Precisión
Artificial1	1	0.9495	10.6652	0.9109	0.9315	0.8927
Artificial2	1	1.8305	12.6711	0.8856	0.9714	0.9024
Artificial3	1	0.5282	8.7624	0.9397	0.9606	0.7485

4.3. Análisis de Resultados

Dadas las características funcionales presentadas en la sección 3.1 y las pruebas realizadas sobre la propuesta, se puede hacer los siguientes análisis:

- En la sección 4.2.1 se presentan una serie de bases de datos con ejemplos etiquetados, en las cuales se aplicó el algoritmo planteado. Se entrenó el modelo, y para cada problema se crearon prototipos que caracterizan las clases. Una vez entrenado el sistema, se evaluó el modelo con un conjunto de prueba, obteniendo resultados satisfactorios para la mayoría de las bases de datos. Se observó que en los casos en los cuales el algoritmo no tiene un buen desempeño, se debe principalmente a la estructura de los datos y al diseño propio de aprendizaje de nuestra propuesta.
- Las pruebas realizadas en la sección 4.2.2 para problemas de clasificación utilizando co-trainning, permite demostrar que la propuesta también puede ser utilizada como un algoritmo semi-supervisado. Además, que el desempeño fue satisfactorio en aquellas bases de datos que cumplen con los criterios establecidos por nuestra propuesta.

- Por otro lado, en la sección 4.2.3 se presentaron pruebas con bases de datos no etiquetadas. Se demostró que el sistema es capaz de crear grupos de manera adecuada siguiendo los criterios planteados en el modelo teórico (sección 3.2.2.1). Además, existen problemas como el de *engyTime* (véase la Figura 4.15) que la propuesta no puede resolver, debido a que viola principalmente el criterio definido para crear grupos en el entrenamiento del modelo; básicamente, la construcción de dos grupos se define si están separados por una distancia "grande" o por baja densidad en sus fronteras.
- Las pruebas dadas en la sección 4.2.4 sobre bases de datos artificiales, permitió mostrar la capacidad del algoritmo de recibir datos etiquetados y no etiquetados en una misma base de datos. La propuesta creó grupos y clases siguiendo los criterios propuestos, logrando combinar armónicamente las tareas de clasificación y agrupamiento.
- Se evidenció en las pruebas sobre bases de datos artificiales, que una vez entrenado el modelo es posible la creación de nuevos prototipos con características completamente diferentes a los prototipos existentes en el sistema, por lo tanto, tiene la capacidad de detectar nuevas agrupaciones que se puedan suscitar debido a nuevas características de los datos de entrada.
- Las métricas de evaluación del proceso de prueba y entrenamiento, en los tres casos planteados (clasificación, agrupamiento e híbrido), permitieron cuantificar la calidad del algoritmo diseñado y validar los resultados obtenidos.

En cuanto a los criterios y condiciones establecidas en nuestro trabajo (ver Sección 3.2.1), se puede afirmar que:

- El criterio (ii), para asignar un ejemplo a un prototipo por medio del umbral de vecindad, es adecuado para la mayoría de las bases de datos; sin embargo, existen estructuras en las cuales este planteamiento no es favorable y es necesario el uso de otro enfoque de MD. El umbral de vecindad no es

favorable en aquellos problemas de clasificación en los que los ejemplos pertenecientes a una clase no son necesariamente parecidos entre sí, están solapados, o tienen alta correlación, como se observó en las pruebas de la sección 4.2.1, principalmente con la base de datos *ios*.

- La principal característica que se puede observar de nuestra propuesta es la forma de crear y separar grupos como parte del proceso de aprendizaje no supervisado, dado por el criterio (iv), en las pruebas realizadas en la sección 4.2.2 se demostró que este criterio se cumple adecuadamente. Sin embargo, este criterio es restrictivo en los problemas en los cuales se puede encontrar grupos solapados, debido a que se realiza la suposición que la base de datos no está compuesta por agrupaciones solapadas, sino más bien, por grupos moderadamente diferenciados entre sí.
- El criterio (vii), que permite la mezcla entre clases, se define bajo condiciones poco probables de ocurrencia, ya que es un problema que no se contempla en los algoritmos de MD, y es más bien delegado a problemas de procesamiento de datos.
- La creación de nuevos prototipos dado por el criterio (v), abarca la detección de datos anómalos, datos con ruido, o nuevos grupos que se están formando, que podrían proporcionar información del problema que el modelo está abordando, como se demostró en los problemas experimentales dados en la sección 4.2.4.

Los criterios propuestos abarcan problemas asociados a agrupamiento y a clasificación, que constituyen una especie de guía metodológica para proceder según sea el escenario que se está evaluando.

Capítulo 5

Conclusiones y Recomendaciones

5.1. Conclusiones

El problema de hibridación de MD presentado es un aporte importante en esta área, ya que permite la detección de nuevos grupos de los cuales puede no tenerse conocimiento, además, permite que en una misma estructura se puedan realizar las tareas de clasificación y agrupamiento. Actualmente, no se cuenta con algoritmos con estas características, ya que los mismos son diseñados para tareas específicas, ya sea de clasificación o de agrupamiento.

Mientras que la mayoría de los algoritmos después de ser entrenados no pueden modificar su estructura, nuestra propuesta es capaz de crear nuevos prototipos después de ser entrenado, detectando así nuevas anomalías en el problema en estudio, dichas anomalías podrían representar: ruido, nuevo conocimiento que se está formando a partir de las entradas al sistema, o nuevas posibles clases que se están integrando al modelo. Los algoritmos clásicos están diseñados para mantener un modelo estático una vez que el modelo ha sido entrenado, en caso de que se deseen integrar nuevas clases se debe entrenar el modelo de nuevo. El aporte de

esta propuesta es que esta actualización se realiza en el funcionamiento en línea, sin necesidad de entrenamientos adicionales. Dicha actualización se puede realizar debido a que el sistema es capaz de la creación de nuevos grupos, estos grupos pueden ser etiquetados como nuevas clases (si se conoce las etiquetas de las entradas) una vez que fueron creados.

El planteamiento se basa en dos enfoques, como lo son el de similaridad con el centroide y 1-NN. Por separado, la similaridad con el centroide genera muchos errores a la hora de asignar un ejemplo a un prototipo, ya que es posible que el centroide no sea completamente una medida representativa de todo el prototipo; por otro lado, 1-NN es más preciso a la hora de realizar la asignación de un ejemplo a una clase/grupo, pero es muy costoso computacionalmente, y no es factible a la hora de evaluar una gran cantidad de datos. Nuestra propuesta integra ambos enfoques de manera tal que toma las ventajas de la similaridad con el centroide (rapidez y bajo costo computacional) y del vecino más cercano (precisión). Dicha combinación permite la asignación parcial de un ejemplo a un prototipo por medio de la similaridad con el centroide, y luego la búsqueda del vecino más cercano dentro de dicho prototipo o su prototipo vecino (no sobre todos los ejemplos de la base de datos, como 1-NN), lo cual reduce considerablemente los tiempos de respuesta.

El objetivo alcanzado por nuestra propuesta es soportar en un mismo sistema las tareas de clasificación pura, agrupamiento puro, y el caso híbrido. Esto representa un gran aporte al área de aprendizaje automático y a todas las áreas que se apoyan en la MD para la extracción de conocimiento. Los resultados obtenidos para clasificación y agrupamiento por separado, al ser comparados con otros algoritmos de cada tarea, fueron satisfactorios. Mientras que para el caso especial híbrido, se

cumplieron todos los objetivos planteados para el estudio, con bases de datos con ejemplos etiquetados y no etiquetados.

5.2. Recomendaciones

Como principal recomendación se plantea la creación bases de datos reales que representen el caso híbrido planteado por el presente proyecto, las mismas deben ser validadas por expertos de diversas áreas, dependiendo del problema asociado.

Un elemento que podría mejorar el desempeño del algoritmo es la asignación del umbral de vecindad, ya que el mismo juega un papel importante a la hora de tomar decisiones para la creación de prototipos; es por ello que se plantea como trabajo futuro la creación de un método adaptativo más eficiente para el cálculo del umbral que el actual. En nuestra propuesta, inicialmente todos los prototipos tienen el mismo umbral y es definido por la varianza de todos los datos. Ahora bien, para evitar la dependencia a ese valor inicial del umbral, se podría plantear un método que inicialice el umbral aleatoriamente o con un valor por defecto, y además, en vez de usar la varianza o distancia promedio entre todos los datos, que busque el umbral óptimo de cada uno de los prototipos por medio de la optimización de una función objetivo, que considere por ejemplo información como la varianza o distancia promedio entre los datos, pero igualmente otros aspectos tales como la correlación, error cuadrático de la distancia, entre otros.

Funcionalmente, 1-NN juega un papel importante en el algoritmo propuesto, es por ello que se plantea la extensión del mismo a K-NN. Este método en particular no escoge el vecino más cercano sino los K vecinos más cercanos (Larose, D., 2004). El valor de K sería proporcionado por el usuario, y depende directamente de la densidad de la base de datos, mientras más densa más grande es K y viceversa (eso da idea también de cómo determinarlo automáticamente, tal que no sea dado por el

usuario). El método de votación de K-NN se modificaría en función al umbral de vecindad de cada prototipo. Es decir, se pensaría cuáles K ejemplos son los vecinos más cercanos a otro ejemplo si tienen la distancia mínima, y además, esa distancia cumple con el umbral de vecindad de su prototipo correspondiente.

Para aplicar el algoritmo a gran variedad de problemas de MD, se plantea la extensión del mismo a base de datos que contengan valores nominales, una de las vías para resolver este problema es la transformación de los valores nominales a valores numéricos a través de la creación de un método de conversión válido.

Finalmente, para ser accesible dicho algoritmo a ingenieros de conocimiento que lo deseen probar o utilizar, se plantea como trabajo fututo la implementación de una interfaz con las funcionalidades necesarias para una adecuada interacción con el usuario.

Referencias Bibliográficas

Adams R. and Fournier J (1975). *Sobolev Spaces*. San Diego, CA, USA: Academic Press.

Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Basu S., Bilenko M. , and Mooney R.J. (2004). A Probabilistic Framework for Semi-Supervised Clustering. Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004). pp. 59-68.

Breiman L. (1996). Bagging Predictors. Machine Learning. Vol. 24, No 2, pp 123-140.

Clark P. and Niblett T. (1989) The CN2 inductuin algorithm. Machine Learning Vol. 3, No 4, pp. 261-283.

Chan P., Fan W., Prodromidis A. and Stolfo S. (1999). Distributed Data Mining in Credit Card Fraud Detection. IEEE Intelligent Systems. Vol 14, No 6, pp.67-74.

Chapelle, O., Scholkopf, B., and Zien, A-Editors (2006). Semi-Supervised Learning. MIT Press, Cambridge, MA.

Cios K. and Kurgan L. (2004) CLIP4: Hybrid inductive machine learning algorithm that generates inequality rules. Information Sciences. Vol .163,No.1, pp. 37-83.

Cios K. and Kurgan L.(2001) Hybrid inductive machine learning: An overview of CLIP algorithms, in: L.C. Jain, J. Kacprzyk (Eds.), New Learning Paradigms in Soft Computing, Physica-Verlag (Springer). pp. 276-322.

Cios K. and Liu N. (1995) An Algorithm which Learns Multiple Covers via Integer Linear Programming. Part I—The CLILP2 Algorithm, Kybernetes. Vol. 24, No 2, pp. 29-50.

Cios K. and Liu N. (1995) An Algorithm which Learns Multiple Covers via Integer Linear Programming. Part II—experimental results and conclusions, Kybernetes. Vol. 24, No. 3, pp. 24-36.

Druck G., Pal C., Zhu X. and McCallum A.(2007) Semi-Supervised Classification with Hybrid Generative/Discriminative Methods, KDD'07, Copyright 2007 ACM 978-1-59593-609-7/07/0008.

Dunham M.(2003) Data Mining: Introductory and Advanced Topics.Prentice Hall. http://lyle.smu.edu/~mhd/book.

Efron, B.; Tibshirani, R. (1993). *An Introduction to the Bootstrap*. Boca Raton, FL: Chapman & Hall/CRC. ISBN 0-412-04231-2.

Fayyad,U., Piatetsky-Shapiro,G., and Smyth,P (1996). The KDD process for Extracting Useful Knowledge from Volumes of Data. Communications of the ACM. Vol.39,No. 11, pp. 51-57.

Fisher R. (1936). The use of multiple measurements in taxonomic problems". *Annals of Eugenics.* Vol. **7** , No. 2, pp. 179-188.

Freund Y and Schapire R.(1999). A Short Introduction to Boosting. Journal of Japanese Society for Artificial Intelligence. Vol. 14, No.5, pp.771-780.

Gabrys B. and Bargiela A. (2000) General Fuzzy Min-Max Neural Network for Clustering and Classification. IEEE Transactions on Neural Networks. Vol. 11, No. 3, pp. 739-783.

Gabrys B. and Petrakieva L.(2002) Combining labelled and unlabelled data in the design of pattern classification systems. In: Hybrid Methods for Adaptive Systems (HMAS'2002) Workshop.

Goldman S. and Zhou Y. (2000) Enhancing Supervised Learning with Unlabeled Data. In Proceedings of the Seventeenth ICML.

Hastie T., Tibshirani R. and Friedman J.(2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Second Edition." Springer Series in Statistics.

He H. and Garcia E. (2009) Learning from Imbalanced Data. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING. Vol. 21, No. 9, pp. 1263–1284.

Intrator N., Reisfeld D. and Yeshurun Y. (1996) Face recognition using a hybrid supervised/unsupervised neural network Pattern Recognition Letters. Vol. 17 pp. 67-76.

Jolliffe I.(2002). Principal Component Analysis. Springer Series in Statistics, 2nd ed. XXIX, ISBN 978-0-387-22440-4.

Karayiannis N. and Weiqun C. (1997) Growing Radial Basis Neural Networks: Merging Supervised and Unsupervised Learning with Network Growth Techniques. IEEE Transactions on Neural Networks. Vol. 8, No. 6, pp. 1492-1506.

Larose D. (2004) *k*-Nearest Neighbor Algorithm, in Discovering Knowledge in Data: An Introduction to Data Mining, John Wiley & Sons, Inc., Hoboken, NJ, USA. doi: 10.1002/0471687545.ch5.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations. Vol 11, No 1, pp.10-18.

Pacheco, F., Rangel, C., Aguilar, J., Cerrada, M. y Altamiranda, J.(2014) Methodological Framework for Data Processing based on the Data Science Paradigm. Proceedings of the XL Conferencia Latinoamericana en Informática (CLEI 2014), IEEE Xplore.

Pedregosa, F. et al (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. Vol 12, pp. 2825-2830.

Pedrycz W., Amato A., Di Lecce V. and Piuri V. (2008) Fuzzy Clustering With Partial Supervision in Organization and Classification of Digital Images. IEEE Transactions on Fuzzy Systems, Vol. 16, No. 4, pp.1008-1026.

Phua C., Alahakoon D. and Lee V. (2004). Minority Report in Fraud Detection: Classification of Skewed Data, SIGKDD Explorations. Vol. 6, No. 1, pp. 50-59.

Rangel, C., Pacheco, F., Aguilar, J., Cerrada, M. y Altamiranda, J.(2013) Methodology for detecting the feasibility of using data mining in an organization, Proceedings of the XXXIX Conferencia Latinoamericana en Informática (CLEI 2013), IEEE Xplore, Vol. 1, pp. 502-513.

Salama K. and Freitas A.(2013) Classification with Clustering-based Bayesian Multi-Nets using Ant Colony Optimization. Evolutionary Computation (CEC), 2013 IEEE Congress on IEEE Xplore. pp. 3079-3086.

Simpson P. (1992) Fuzzy Min-Max Neural Networks- Part 1: Classification. IEEE Transactions on Neural Networks. Vol. 3, pp. 776-786.

Simpson P. (1993) Fuzzy Min-Max Neural Networks- Part 2: Clustering. IEEE Transactions on Fuzzy Systems, Vol. 1, No. 1, pp. 32-45.

Sokolova M. and Lapalme G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing and Management. Vol.45, No. 4, pp. 427-437.

Spiegel, M. (1992) Theory and Problems of Probability and Statistics, 2nd ed. New York: McGraw-Hill, pp. 294-323.

Takashi W. and Hiroshi M. (2003) State of the art of graph-based data mining. ACM SIGKDD Explorations Newsletter. Vol. 5, No. 1, pp. 59-68.

Tan P., Steinbach M. and Kumar V (2005). Introduction to Data Mining (First Edition) Addison Wesley. http://www-users.cs.umn.edu/~kumar/dmbook/index.php.

Ultsch, A. (2005) Clustering with SOM: U*C, In *Proc. Workshop on Self-Organizing Maps.* pp. 75-82.

Verleysen M. y Damien F (2005). The Curse of Dimensionality in Data Mining and Time Series Prediction. IWANN, LNCS 3512. pp. 758 -770.

Witten I. and Frank E. (2005) Data Mining: Practical Machine learning Tools and Techniques, Second Edition. Editorial Elsevier.

Xu R. and Wunsch D.(2005) Survey of Clustering Algorithms. IEEE Transactions on Neural Networks. Vol. 16, No. 3, pp 645-678.

Yahong H., Yi Y., Yan Y., Zhigang M., Nicu S. and Xiaofang Z. (2015). Semisupervised Feature Selection via Spline Regression for Video Semantic Recognition. IEEE Transactions on Neural Networks and Learning Systems. Vol. 26, No. 2, pp. 252-264.

Yang Q. and Wu X.(2006) 10 Challenging Problems in Data Mining Research. International Journal of Information Technology & Decision Making. Vol. 5, No. 4, pp. 597-604.

Zhao Y. and Zhang Y. (2000). Comparison of decision tree methods for finding active objects. Advances in Space Research. Vol. 42, No.12, pp. 1955-1959.

Apéndices

Apéndice A

A continuación, se describen las clases y las funciones más resaltantes implementadas para la construcción del algoritmo.

1. Instancia

   El objeto instancia está definida por un ejemplo en la base de datos de MD, por lo tanto está compuesta por los atributos y la etiqueta, como se muestra en la Figura A.1. La funciones para esta clase son las clásicas de set's y get's, adicionalmente hay una función que verifica si el ejemplo es etiquetado o no.

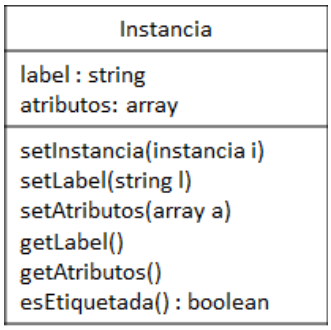


Figura A. 1. Diagrama de clase del objeto Instancia

2. Instancias

   Esta clase está conformada principalmente por una lista de instancias, además de un diccionario que tiene almacenada la etiqueta y la cantidad de instancias que tienen esta etiqueta. Las funciones principales implementadas se muestran en la Figura A.2.

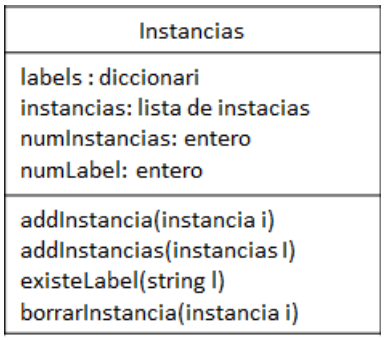


Figura A.2. Diagrama de clase del objeto Instancias.

3. Data

Esta clase es vital para el funcionamiento del sistema, ya que en ella se realiza la lectura del archivo donde se encuentra la base de datos, Las funciones principales implementadas se muestran en la Figura A.3.

En esta clase se cargan los datos de acuerdo a la extensión del archivo, dichos datos son cargados en las estructuras aceptadas por el algoritmo definido por instancias. Las funciones más resaltantes de esta clase son:

a. *CargarData (string file):* carga los datos dada la dirección del archivo, el formato permitido del archivo para el uso de esta función esta .txt y .arff.
b. *CargarData (array data):* carga los datos dado un arreglo *numpyarray* con los datos
c. *CargarAtributos (array data):* carga los datos dada un arreglo *numpyarray* con los atributos
d. *CargarLabels (string file):* carga las etiquetas de los datos dada la dirección del archivo.

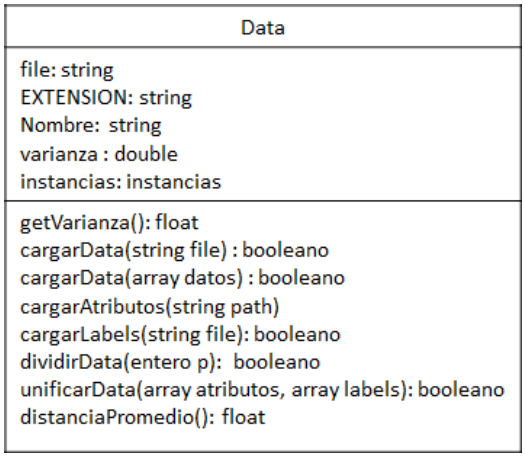


Figura A.3. Diagrama de clase del objeto Data.

4. Prototipo

Esta clase representa los grupos y clases creados por el algoritmo, está conformado por las instancias que pertenecen al prototipo, el umbral de vecindad del mismo, el centroide del prototipo y otros atributos que completan la información necesaria para definir un prototipo (véase la Figura A.4).

Las funciones más resaltantes de esta clase son:

a. *addInstancia(instancia i) :* integra una nueva instancia al prototipo y actualiza todos sus parámetros como el centroide.
b. *deleteInstancia(instancia i) :* elimina una instancia del prototipo y actualiza todos sus parámetros como el centroide.
c. *mezclar(Prototipo p)*: mezcla el prototipo actual con el prototipo p
d. *actualizarUmbral() :* esta función permite actualizar el umbral de vecindad.

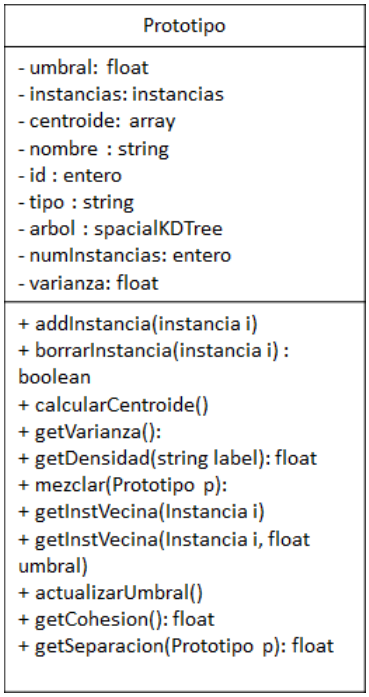


Figura A.4. Diagrama de clase del objeto Prototipo.

5. Algoritmo:

Esta clase es vital para el funcionamiento del prototipo, ya que permite el entrenamiento del algoritmo y el funcionamiento en línea del mismo. Está conformado por todas las funciones necesarias para la implementación del algoritmo dado en el Capítulo 3 y está representada por el diagrama de la Figura A.5.

Esta clase está compuesta principalmente por:

a. *construirModelo(Instancias I)*: permite entrenar el algoritmo con los datos de entrenamiento.
b. *actualizarPrototipos()*: realiza la modificación necesaria de los prototipos para cumplir las reglas dadas en el Capítulo 3.
c. *actualizarPrototipos()*: realiza la modificación necesaria de los prototipos basado en las áreas densas.
d. *funcinamientoLinea(Instancia i):* dada una instancia verifica a que prototipo pertence la misma y la agrega a dicho prototipo.
e. *similaridad(instancia i)* : genera una lista con la similaridad de la instancia i a todos los prototipos existentes.

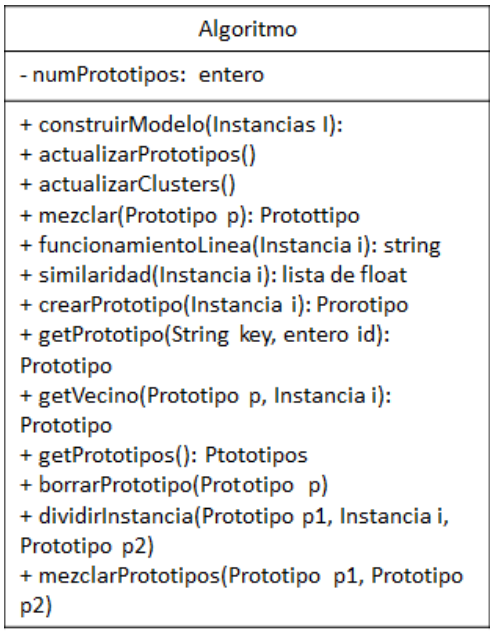


Figura A.5. Diagrama de clase del objeto Algoritmo.

6. Evaluación:

Esta clase permite evaluar los prototipos generados en la clase algoritmo, existen diferentes tipos de evaluaciones dependiendo del problema que se

está tratando. Así mismo, se implementaron otras funciones para obtener métricas de evaluación conocidas como se evidencia en la Figura A.6.

a. *Evaluar (Prototipos p)*: Calcula la precisión, error, clasificados correctamente e incorrectamente en el caso de clasificación. En el caso de clustering calcula la cohesión y separación de los clusters. Genera un resumen por pantalla de los resultados obtenidos.
b. *EvaluarTest(array label,array labelTest)*: Esta función solo es válida para problemas de clasificación. Recibe el resultado obtenido con el algoritmo propuesto una vez evaluado un conjunto de prueba, y lo compara con sus etiquetas originales. Se calculan todas las métricas de clasificación y se genera un resumen por pantalla.
c. *EvaluarSemi(Prototipos p, array data, array labels):* Esta función solo es válida para problemas de clasificación tratados con el enfoque semi-supervisado. Permite verificar que los ejemplos no etiquetados sean asignados a las clases con sus etiquetas originales. Genera un resumen de los resultados obtenidos.

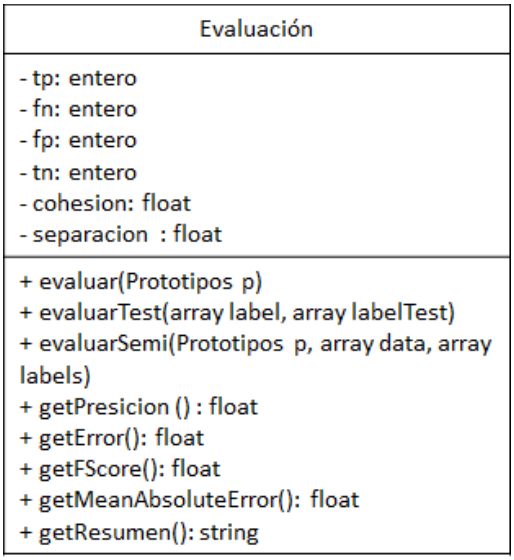


Figura A.6. Diagrama de clase del objeto Evaluación.

7. PCAadatpatado:

Permite graficar tanto los datos originales como los prototipos generados con el algoritmo utilizando PCA. El uso de PCA se realiza por medio de una librería de sci-learn y se adapta a nuestra propuesta, el diagrama viene dado en la Figura A.7.

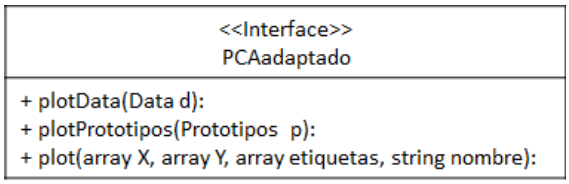


Figura A.7. Diagrama de clase del objeto PCAadaptado.